Master M2DS-SAF: TP Optimisation

Algorithme Forward-Backward (gradient proximal)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
# from dataclasses import dataclass, field
```

On souhaite programmer et tester un algorithme permettant de minimiser par rapport à x:

$$\frac{1}{2}\|\mathrm{b} - \mathrm{Ax}\|_2^2 + \lambda\|\mathrm{x}\|_1$$

où sont donnés $\mathrm{b} \in \mathrm{R}^{\mathrm{m}}$, $\mathrm{A} \in \mathrm{R}^{\mathrm{m}\times\mathrm{n}}$ et $\lambda > 0$. On notera qu'il s'agit de l'estimateur dit "LASSO".

On utilisera l'algorithme du gradient proximal (ou Forward-Backward). Programmer cet algorithme.

In [2]:
```
def soft_thresh(u, s):
    """ v = soft_thresh(u, s)

    Returns a soft-thresholding (elementwise) of u with threshold s
    v = sign(v)*maximum(0, abs(u) - s)
    """
    v = np.sign(u)*np.maximum(0, abs(u) - s)
    return v


def lasso(A, b, lamb, maxiter=5000):
    """ lasso(A, b, lamb, maxiter=500)

    min. 1/2*norm(b- Ax)^2+lambda*sum(abs(x))
    (wrt x)
    Forward-Backward algorithm

    Parameters
    ----------
    A : array, shape (T, T)
    b : array, shape (T,) or (T, 1)
    lamb : float
    maxiter : int (default=5000)

    Returns
    -------
    x : array, shape (T,)
    """
    b = b.squeeze()
    n = A.shape[1]
    gam = 1.9/np.linalg.norm(A, 2)**2

    x = np.zeros(n)
#    history = {'crit':[]}
    prec = 1e-6
    crit = 0
    for nit in range(maxiter):
        Ax = A.dot(x)
        critold = crit
        crit = np.linalg.norm(Ax-b)**2/2+lamb*sum(abs(x))
#        history['crit'].append(crit)
        if nit > 1 and critold-crit < prec*critold:
            break
```

```
        x = x-gam*A.transpose().dot(Ax-b)
        x = soft_thresh(x, lamb*gam)
#         print('{0} crit = {1}\n'.format(nit, crit))
    # print('   Exiting at iteration {0} (maxiter = {1}).\n'
    # .format(nit, maxiter))
    return x
```

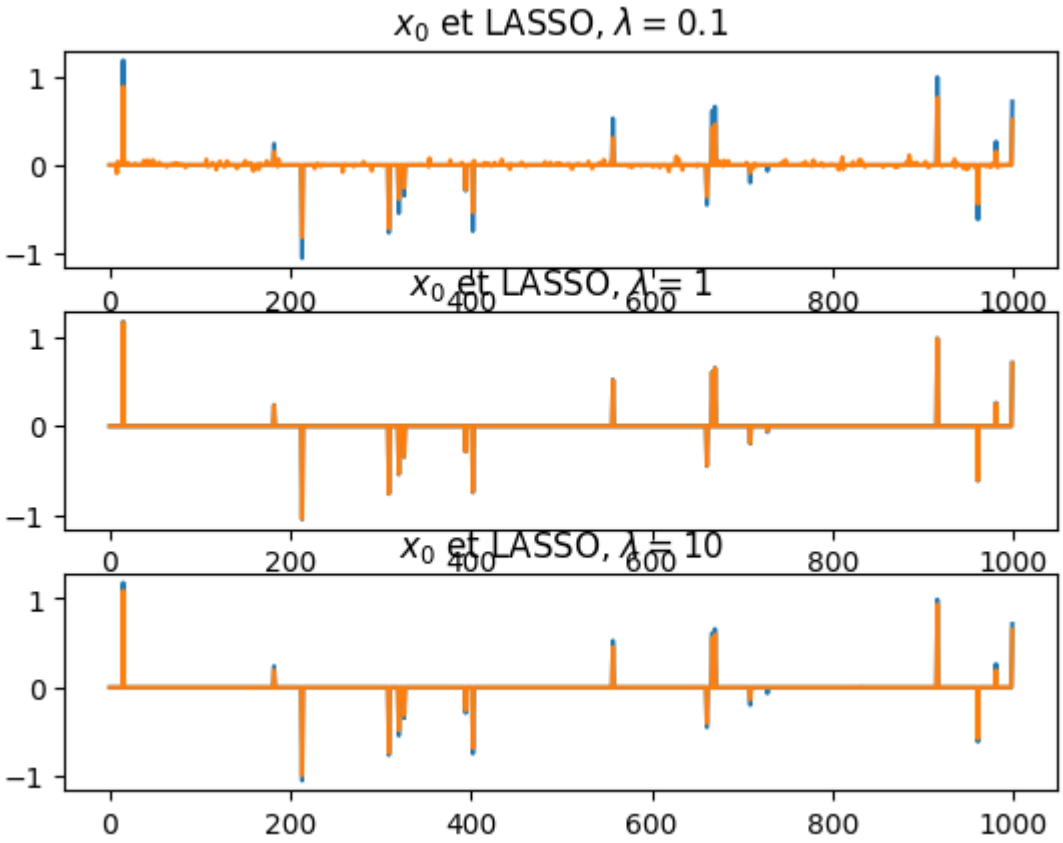
Tester l'estimation LASSO dans le cas où $m = 200, n = 1000$, $x_0 \in R^n$ est un vecteur parcimonieux (2% coefficients non nuls) et $b \in R^m$ est un vecteur d'observations bruitées $b = Ax_0 + b$ avec $A \in R^{m \times n}$. Comparer l'estimateur du LASSO à x_0 pour différentes valeurs de λ.

In [3]:
```
# génération des données
m, n = 200, 1000
s = 0.02

A = np.random.randn(m, n)
x0 = np.asarray([0 if np.random.rand() > s else np.random.randn()
                 for i in range(n)])
y0 = A@x0

sigma = 0
e = sigma*np.random.randn(m)
yn = y0 + e
```

In [4]:
```
# calcul du LASSO pour differents lambda
lamb1, lamb2, lamb3 = 0.1, 1, 10
xlasso1 = lasso(A, yn, lamb1)
xlasso2 = lasso(A, yn, lamb2)
xlasso3 = lasso(A, yn, lamb3)
```

In [5]:
```
# Tracés
FigLasso, AxLasso = plt.subplots(3, 1, num=3)
AxLasso[0].plot(x0)
AxLasso[0].plot(xlasso1)
AxLasso[0].set_title('$x_0$ et LASSO, $\lambda=$'+str(lamb1))
AxLasso[1].plot(x0)
AxLasso[1].plot(xlasso2)
AxLasso[1].set_title('$x_0$ et LASSO, $\lambda=$'+str(lamb2))
AxLasso[2].plot(x0)
AxLasso[2].plot(xlasso3)
AxLasso[2].set_title('$x_0$ et LASSO, $\lambda=$'+str(lamb3))
```

Out[5]:
```
Text(0.5, 1.0, '$x_0$ et LASSO, $\\lambda=$10')
```


x_0 et LASSO, $\lambda = 0.1$
x_0 et LASSO, $\lambda = 1$
x_0 et LASSO, $\lambda = 10$
1
0
−1
0
200
400
600
800
1000

In [ ]: