David J. Elkind - LIPO

I'm interested in estimating the Lipschitz constant k for functions which approximate as Lipschitz. When the functions are noisy, they can't be Lipschitz, but adding a noise term s can make the functions behave more nicely; including the large penalty 10^6 for s forces it to often be 0.

The function of interest is f and it is only observed via some finite number of points $(x_i, f(x_i))$ for $i = 1, 2, \ldots t$. Define

$$U(y) = \min_{i \in 1,2,\ldots t} \left[f(x_i) + \sqrt{s_i + (y - x_i)^\top K (y - x_i)} \right] \tag{1}$$

One way to estimate k is to optimize the following program (using a different k_i for each coordinate):

$$\begin{aligned}
\min_{K,s} \quad & ||K||_F^2 + 10^6 \sum_{i=1}^{t} s_i^2 && (2)\\
\text{s.t.} \quad & U(x_i) \geq f(x_i) i \in \{1, 2, \ldots t\} && (3)\\
& s_i \geq 0 \forall i \in \{1, 2, \ldots t\} && (4)\\
& K_{i,j} \geq 0 \forall i, j \in \{1, 2, \ldots d\}^2 && (5)\\
& \text{K is a diagonal matrix.} && (6)
\end{aligned}$$

This clearly can be rewritten as

$$\begin{aligned}
\min_{k,s} \quad & k^\top k + 10^6 \cdot s^\top s && (7)\\
\text{s.t.} \quad & U(x_i) \geq f(x_i) \ i \in \{1, 2, \ldots t\} && (8)\\
& s_i \geq 0 \ \forall i \in \{1, 2, \ldots t\} && (9)\\
& k_i \geq 0 \ \forall i \in \{1, 2, \ldots d\}^2 && (10)
\end{aligned}$$

But it's not at all clear how to me actually *do* the optimization, since the minimization objective is not obviously a QP. I guess one way to do it would be to roll the two summands together and write

$$\begin{aligned}
\min_{k,s} \quad & \begin{bmatrix} k \\ s \end{bmatrix}^\top A \begin{bmatrix} k \\ s \end{bmatrix} && (11)\\
\text{s.t.} \quad & U(x_i) \geq f(x_i) \ i \in \{1, 2, \ldots t\} && (12)\\
& s_i \geq 0 \ \forall i \in \{1, 2, \ldots t\} && (13)\\
& k_i \geq 0 \ \forall i \in \{1, 2, \ldots d\}^2 && (14)
\end{aligned}$$

Just for convenience, we can rewrite U in terms of k:

$$U(y) = \min_{i \in 1,2,\ldots t} \left[f(x_i) + \sqrt{s_i + ||k^\top (y - x_i)||_2^2} \right] \tag{15}$$

and observe that all I've done is move stuff around so that A is given by

$$A_{ij} = \begin{cases} 1 & \forall i = j \wedge 1 \leq i \leq d \\ 10^6 & \forall i = j \wedge d < i \leq t + d \\ 0 & \text{otherwise} \end{cases}, \tag{16}$$

i.e. A is a diagonal matrix with 1s corresponding to the elements of k and 10^6 corresponding to the elements of s.

This is "nicer" in a sense, since it's *obviously* a QP. But what's less-nice is that U, and therefore its corresponding constraint, is not a linear function, owing to the fact that s and k both appear inside of the square root.

So given that this program is not a QP with linear constraints, what would the appropriate tool be to solve this problem? More specifically, what python library would you recommend using?