In this homework, I set the matrix A:

2 3

1 2

3 4

b:

1

2

3

P1:

1 1

1 3

P2:

2 0

0 1

P3:

2 1

1 2

e = 1 and $\kappa = 0.2$.

To decide μ and L, I take the second derivative both objective function and constraint function. For f_0, the second derivative is: $2A^TA$. For f_1 to f_3, the second derivative is $2P_i^{-1}$. After computing, I choose the minimum of the above as μ and the maximum as L for our problem.

Then I set the function f_0, f_1, f_2 and f_3 and used the "diff" function to calculate their derivative

expression.

Before the loop, I set initial t as -100.

In each inner loop, I first check if $f^*(t; x_k; \mu)$ is larger than $(1-\kappa)$ $f^*(t; x_k; L)$, calculating with Optimization tool "fmincon()". If it is larger, then the condition breaks the inner loop and begins to update the t value. Or the inner loop would keep generating a sequence of x_k until it meets the requirements. With the valid x_k sequence, we find the x_k with minimum $f^*(t; x_k; L)$. This x_k would be our initial point in the next iteration of exterior loop.

In the exterior loop, I use the bidirectional search to find the next t: first check if the current $f^*(t; x_k; \mu)$, where x_k is the last of the inner loop sequence, is between the value when t = -100 (lower boundary) and when t = norm(b) (upper boundary). Then we set

t = (lower boundary + upper boundary) / 2 and check the $f^*(t; x_k; \mu)$ with the new t. if it is larger than zero, the new t becomes the lower boundary. If it is smaller than zero, the new t becomes the upper boundary. As the lower boundary and upper boundary merge together, I can get a t that can make $f^*(t; x_k; \mu)$ as 0 approximately.

In each inner loop iteration, I check if $f^*(t; x_k; L) <= \varepsilon$. If it meets the requirement, the inner loop will save the current x and jump out both inner loop and exterior loop.

The program runs relatively fast (3 seconds). The final result I get is x = [0.6363; 0.2126].