Possible extension to distributed MPC

Let just consider x_p and x_q at a particular time step. For simplicity, let me remove the time index and the bar notation. Also,

$$\sum_{q\in\mathcal{N}_p}(x_q-x_p)^T Q(x_q-x_p) = \sum_{q\in\mathcal{N}_p}\left(x_p^T Q x_p + -2x_q^T Q x_p + x_q^T Q x_q\right) \tag{1}$$

$$= (\#\mathcal{N}_p)x_p^T Q x_p - 2\left[\sum_{q\in\mathcal{N}_p} x_q^T Q\right]^T x_p + \sum_{q\in\mathcal{N}_p} x_q^T Q x_q \tag{2}$$

Letting $A=(\#\mathcal{N}_p)Q$, $b=-2\left[\sum_{q\in\mathcal{N}_p} x_q^T Q\right]^T$ and $c=\sum_{q\in\mathcal{N}_p} x_q^T Q x_q$, we now have the form of

$$x_p^T A x_p + b x_p + c$$

We have N (number of waypoints of trajectory) of this. Note that you can do the same thing for P(for terminal cost) to make A_f, b_f, c_f. But for simplicity, let us just consider $P=Q$. Let me denote $X := [x_{p_1}^T, \dots x_{p_N}^T]^T$ (the path) and input sequence $U := [u_{p_1}^T, \dots, u_{p_N}^T]^T$. Also, let me write $S := [X^T, U^T]^T$. Then the total objective function $J(X,U)$ will be

$$J(X,U) = \sum_{k\in[N]}\left(x_{p_k}^T A x_{p_k} + b x_{p_k} + c\right) + \sum_{k\in[N]} u_{p_k}^T Q u_{p_k} \tag{3}$$

$$= \left(X^T \tilde{A} X + \tilde{b} X + \tilde{c}\right) + U^T \tilde{R} U \tag{4}$$

$$= S^T \mathrm{diag}(\tilde{A}, \tilde{R}) S + [\tilde{b}, \mathrm{zeros}]^T S + \tilde{c} \tag{5}$$

$$\tag{6}$$

wher $\tilde{A} = \mathrm{diag}(A, \dots, A)$, $\tilde{B} = [b, \dots, b]$, $\tilde{c} = \sum_{i\in[N]} c$ and $\tilde{R} = \mathrm{diag}(R, \dots, R)$.

In my original implementation, the cost J is expressed by $S^T H S$ and H is constructed by the following:

```
function H = construct_costfunction(obj)
    % compute H
    Q_block = [];
    R_block = [];
    for itr=1:obj.N
        Q_block = blkdiag(Q_block, obj.sys.Q);
        R_block = blkdiag(R_block, obj.sys.R);
    end
    H = blkdiag(Q_block, obj.sys.P, R_block);
end
```

So first you should replace Q_block and R_block in final line by $\tilde{Q}$ and $\tilde{R}$. As for $obj.sys.P$ you can do the same. Note that in this setting, we have $[\tilde{b}, \mathrm{zeros}]^T S$ and $tildec$. So, just compute then in the constructor when you make 'OptimalControler' instance and then pass them to 'quadprog' (MATLAB's quadprog accept the form of $x^T A x + b^T x$).