Matlab interface for HPIPM

Nikolce Murgovski

The solver HPIPM targets a QP problem in the form

$$
\begin{aligned}
\min_{x,u,s} \quad & \sum_{n=1}^{N} \left(\frac{1}{2}(u_n^T R_n u_n + x_n^T Q_n x_n) + x_n^T S_n u_n + r_n^T u_n + q_n^T x_n \right) + x_f^T Q_f x_f + q_f^T x_f && f = N+1 \\
& + \sum_{n=1}^{N+1} \left(\frac{1}{2}(s_{\mathrm{l}n}^T Z_{\mathrm{l}n} s_{\mathrm{l}n} + s_{\mathrm{u}n}^T Z_{\mathrm{u}n} s_{\mathrm{u}n}) + z_{\mathrm{l}n}^T s_{\mathrm{l}n} + z_{\mathrm{u}n}^T s_{\mathrm{u}n} \right) \\
\text{s.t.} \quad & x_{n+1} = A_n x_n + B_n u_n + b_n && n = 1, \ldots, N \\
& \mathrm{lb}_n - \begin{bmatrix} \\ s_{\mathrm{l}n} \end{bmatrix} \leq \begin{bmatrix} u_n \\ x_n \end{bmatrix} \leq \mathrm{ub}_n + \begin{bmatrix} \\ s_{\mathrm{u}n} \end{bmatrix} && n = 1, \ldots, N \\
& \mathrm{lb}_n - \begin{bmatrix} \\ s_{\mathrm{l}n} \end{bmatrix} \leq \begin{bmatrix} \\ x_n \end{bmatrix} \leq \mathrm{ub}_n + \begin{bmatrix} \\ s_{\mathrm{u}n} \end{bmatrix} && n = N+1 \\
& \mathrm{lg}_n \leq C_n x_n + D_n u_n \leq \mathrm{ug}_n && n = 1, \ldots, N \\
& \mathrm{lg}_f \leq C_f x_f \leq \mathrm{ug}_f && f = N+1 \\
& s = \begin{bmatrix} s_{\mathrm{l}n} \\ s_{\mathrm{u}n} \end{bmatrix} \geq 0 && n = 1, \ldots, N+1
\end{aligned}
$$

where x are states, u are control inputs, N is the number of stages, n_x is the number of states, n_u is the number of control signals and s are slack variables for the soft box constraints on the states. The problem can be solved by running

```
[x,u,lam,lamf,s,lams]=hpipm(N,nx,nu,ng,ngf,x0,A,B,b,Q,Qf,R,S,q,qf,r,
                            lb,ub,C,D,lg,ug,Cf,lgf,ugf,ixb,Zl,Zu,zl,zu,ixs);
```

Output of the function are the states, the control inputs, the lagrange multipliers, lam, the lagrange multipliers in the final stage, lamf, the slack variables and the lagrange multipliers, lams, related to constraints involving the slack variables. The matrices that are input arguments in HPIPM are obtained by horizontal concatenation, e.g.,

$$A = \left[A_1, A_2, ..., A_N\right] \in \mathbb{R}^{n_x \times n_x N}.$$

Alternatively, if constraints are to be kept hard, i.e. $s = 0$, then the problem can be solved by running

```
[x,u,lam,lamf]=hpipm(N,nx,nu,ng,ngf,x0,A,B,b,Q,Qf,R,S,q,qf,r,
                     lb,ub,C,D,lg,ug,Cf,lgf,ugf,ixb);
```

The box bounds lb and ub are constructed by assigning the first nu rows for the bounds on u and the last n_x rows for the bounds on x. The activation of box constraints in individual stages can be controlled by the logical matrix ixb. In order to reduce computation time, the initial state value is not considered an optimization variable with imposed equality constraint to x0. Instead, the function automatically alters the initial bounds lb_0 and ub_0 and the vectors b_0 and r_0, such that optimization variable for the initial state is not needed. Similarly, the logical matrix ixb is automatically set to false for the state box constraints in the first stage, i.e. the last n_x rows. Since final stage cannot have box constraints on the control inputs, ixb is also automatically set to false for the first n_u rows in the final stage.

The first n_x rows in the matrix lam contain the lagrange multipliers for the state dynamics. The next $2(n_u + n_x)$ rows contain the lagrange multipliers for the lower and upper box constraints on the control inputs and the states, respectively. The remaining $2n_g$ rows contain the lagrange multipliers for the lower and upper general constraints. The rows of lamf are composed in a similar way. The first n_x rows in lams are the lagrange multipliers for the slack variables of the

lower box constraints on the states, while the remaining n_x rows are lagrange multipliers for the slack variables of the upper box constraints on the states. The logical matrix ixs is used to indicate which box state constraints are soft. This matrix is automatically altered in accordance to ixb, as soft constraints can only be applied to used box constraints on states. In places where box constraints or slack variables are not used, i.e. where ixb and ixs are false, the function returns not-a-number (NaN) for the lagrange multipliers.

Table 1: Arguments in HPIPM.

Inputs	Size	Description
N	1×1	Number of stages.
nx	1×1	Number of states in a stage.
nu	1×1	Number of control inputs in a stage.
ng	1×1	Number of general constraints in a stage.
ngf	1×1	Number of general constraints in the final stage.
x0	nx $\times$ (N+1)	Initial state values.
A	nx $\times$ nx·N	State matrix in linear dynamics.
B	nx $\times$ nu·N	Control matrix in linear dynamics.
b	nx $\times$ N	Disturbance in linear dynamics.
Q	nx $\times$ nx·N	Quadratic state penalty.
Qf	nx $\times$ nx	Quadratic state penalty in the final stage.
R	nu $\times$ nu·N	Quadratic penalty for control inputs.
S	nx $\times$ nu·N	Cross-term penalty (typically not used).
q	nx $\times$ N	Linear state penalty.
qf	nx $\times$ 1	Linear state penalty in the final stage.
r	nu $\times$ N	Linear penalty for control inputs.
lb	(nu+nx) $\times$ (N+1)	Lower box constraints on control inputs and states.
ub	(nu+nx) $\times$ (N+1)	Upper box constraints on control inputs and states.
C	ng $\times$ nx·N	State matrix in general constraints.
D	ng $\times$ nu·N	Control matrix in general constraints.
lg	ng $\times$ N	Lower bound in general constraints.
ug	ng $\times$ N	Upper bound in general constraints.
Cf	ngf $\times$ nx	State matrix in general constraints in the final stage.
lgf	ngf $\times$ 1	Lower bound in general constraints in the final stage.
ugf	ngf $\times$ 1	Upper bound in general constraints in the final stage.
ixb	(nu+nx) $\times$ (N+1)	Logical matrix indicating which box constraints are active.
Zl	nx $\times$ nx·(N+1)	Quadratic penalty for the slack variables of the lower box constraints.
Zu	nx $\times$ nx·(N+1)	Quadratic penalty for the slack variables of the upper box constraints.
zl	nx $\times$ (N+1)	Linear penalty for the slack variables of the lower box constraints.
zu	nx $\times$ (N+1)	Linear penalty for the slack variables of the upper box constraints.
ixs	nx $\times$ (N+1)	Logical matrix indicating which box state constraints are soft.
Outputs		
x	nx $\times$ (N+1)	Optimal state values.
u	nu $\times$ N	Optimal control values.
lam	(nx+2(nu+nx)+2ng)$\times$N	Lagrange multipliers.
lamf	(nx+2(nu+nx)+2ngf)$\times$1	Lagrange multipliers in the final stage.
s	2 nx $\times$ (N+1)	Slack variables.
lams	2 nx $\times$ (N+1)	Lagrange multipliers for the slack variables.