HPIPM reference guide

Gianluca Frison

July 2, 2019

Contents

Chapter 1

Introduction

HPIPM, which stands for High-Performance Interior Point Method, is a library providing a collection of quadratic programs (QP) and routines to manage them. Aim of the library is to provide both stand-alone IPM solvers for the QPs and the building blocks for more complex optimization algorithms.
At the moment, three QPs types are provided: dense QPs, optimal control problem (OCP) QPs, and tree-structured OCP QPs. These QPs are defined using C structures. HPIPM provides routines to manage the QPs, and to convert between them.
HPIPM is written entirely in C, and it builds on top of BLASFEO [1], that provides high-performance implementations of basic linear algebra (LA) routines, optimized for matrices of moderate size (as common in embedded optimization).

Chapter 2

Dense QP

The dense QP is a QP in the form

$$
\begin{aligned}
\min_{v,s} \quad & \frac{1}{2}\begin{bmatrix} v \\ 1 \end{bmatrix}^T \begin{bmatrix} H & g \\ g^T & 0 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} s^l \\ s^u \\ 1 \end{bmatrix}^T \begin{bmatrix} Z^l & 0 & z^l \\ 0 & Z^u & z^u \\ (z^l)^T & (z^u)^T & 0 \end{bmatrix} \begin{bmatrix} s^l \\ s^u \\ 1 \end{bmatrix} \\
\text{s.t.} \quad & Av = b, \\
& \begin{bmatrix} \underline{v} \\ \underline{d} \end{bmatrix} \leq \begin{bmatrix} J_{b,v} \\ C \end{bmatrix} v + \begin{bmatrix} J_{s,v} \\ J_{s,g} \end{bmatrix} s^l, \\
& \begin{bmatrix} J_{b,v} \\ C \end{bmatrix} v - \begin{bmatrix} J_{s,v} \\ J_{s,g} \end{bmatrix} s^u \leq \begin{bmatrix} \overline{v} \\ \overline{d} \end{bmatrix}, \\
& s^l \geq \underline{s}^l, \\
& s^u \geq \underline{s}^u,
\end{aligned}
$$

where v are the primal variables, s^l (s^u) are the slack variables of the soft lower (upper) constraints. The matrices $J_{\dots}$ are made of rows from identity matrices. Furthermore, note that the constraint matrix with respect to v is the same for the upper and the lower constraints.

Chapter 3

OCP QP

The OCP QP is a QP in the form

$$
\begin{aligned}
\min_{x,u,s} \quad & \sum_{n=0}^{N} \frac{1}{2} \begin{bmatrix} u_n \\ x_n \\ 1 \end{bmatrix}^T \begin{bmatrix} R_n & S_n & r_n \\ S_n^T & Q_n & q_n \\ r_n^T & q_n^T & 0 \end{bmatrix} \begin{bmatrix} u_n \\ x_n \\ 1 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} s_n^l \\ s_n^u \\ 1 \end{bmatrix}^T \begin{bmatrix} Z_n^l & 0 & z_n^l \\ 0 & Z_n^u & z_n^u \\ (z_n^l)^T & (z_n^u)^T & 0 \end{bmatrix} \begin{bmatrix} s_n^l \\ s_n^u \\ 1 \end{bmatrix} \\
\text{s.t} \quad & \\
& x_{n+1} = A_n x_n + B_n u_n + b_n, && n = 0, \dots, N-1, \\
& \begin{bmatrix} \underline{u}_n \\ \underline{x}_n \\ \underline{d}_n \end{bmatrix} \leq \begin{bmatrix} J_{b,u,n} & 0 \\ 0 & J_{b,x,n} \\ D_n & C_n \end{bmatrix} \begin{bmatrix} u_n \\ x_n \end{bmatrix} + \begin{bmatrix} J_{s,u,n} \\ J_{s,x,n} \\ J_{s,g,n} \end{bmatrix} s_n^l, && n = 0, \dots, N, \\
& \begin{bmatrix} J_{b,u,n} & 0 \\ 0 & J_{b,x,n} \\ D_n & C_n \end{bmatrix} \begin{bmatrix} u_n \\ x_n \end{bmatrix} - \begin{bmatrix} J_{s,u,n} \\ J_{s,x,n} \\ J_{s,g,n} \end{bmatrix} s_n^u \leq \begin{bmatrix} \overline{u}_n \\ \overline{x}_n \\ \overline{d}_n \end{bmatrix}, && n = 0, \dots, N, \\
& s_n^l \geq \underline{s}_n^l, && n = 0, \dots, N, \\
& s_n^u \geq \underline{s}_n^u, && n = 0, \dots, N,
\end{aligned}
$$

where u_n are the control inputs, x_n are the states, s_n^l (s_n^u) are the slack variables of the soft lower (upper) constraints and $\underline{s}_n^l$ and $\underline{s}_n^u$ are the lower bounds on lower and upper slacks, respectively. The matrices $J_{\dots,n}$ are made of rows from identity matrices. Note that all quantities can vary stage-wise. Furthermore, note that the constraint matrix with respect to u and x is the same for the upper and the lower constraints.

```
int d_memsize_ocp_qp(int N, int *nx, int *nu, int *nb, int *ng, int *ns);

void d_create_ocp_qp(int N, int *nx, int *nu, int *nb, int *ng, int *ns,
    struct d_ocp_qp *qp, void *memory);

void d_cvt_colmaj_to_ocp_qp(double **A, double **B, double **b,
    double **Q, double **S, double **R, double **q, double **r,
    int **idxbx, double **lbx, double **ubx,
    int **idxbu, double **lbu, double **ubu,
    double **C, double **D, double **lg, double **ug,
    double **Zl, double **Zu, double **zl, double **zu, int **idxs,
    double **ls, double **us,
    struct d_ocp_qp *qp);
```

Bibliography

[1] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl. BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 2018. (accepted).