ACADOS / ACADO 2.0 / ???
Reference

February 11, 2016

Chapter 1

Interfaces

1.1 OCP QP Interface

The interface describes an Optimal Control Problem (OPC) Quadratic Programming (QP) problem in the form

$$\min_{x,u} \quad \sum_{n=0}^{N-1} \frac{1}{2} \begin{bmatrix} u_n \\ x_n \\ 1 \end{bmatrix}^T \begin{bmatrix} R_n & S_n & r_n \\ S_n^T & Q_n & q_n \\ r_n^T & q_n^T & \end{bmatrix} \begin{bmatrix} u_n \\ x_n \\ 1 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} x_N \\ 1 \end{bmatrix}^T \begin{bmatrix} Q_N & q_N \\ q_N^T & \end{bmatrix} \begin{bmatrix} x_N \\ 1 \end{bmatrix} \tag{1.1}$$

$$s.t. \quad x_{n+1} = A_n x_n + B_n u_n + b_n \quad , \quad n = 0, \ldots, N-1 \tag{1.2}$$

$$\underline{u}_n \leq u_n \leq \overline{u}_n \quad , \quad n = 0, \ldots, N-1 \tag{1.3}$$

$$\underline{x}_n \leq x_n \leq \overline{x}_n \quad , \quad n = 0, \ldots, N \tag{1.4}$$

$$\underline{d}_n \leq C_n x_n + D_n u_n \leq \overline{d}_n \quad , \quad n = 0, \ldots, N-1 \tag{1.5}$$

$$\underline{d}_N \leq C_N x_N \leq \overline{d}_N \tag{1.6}$$

The C code interface looks like

```
int ocp_qp_SOLVERNAME(
    int N, int *nx, int *nu, int *nb, int *ng,
    double **A, double **B, double **b,
    double **Q, double **S, double **R, double **q, double **r,
    int **idxb, double **lb, double **ub,
    double **C, double **D, double **ld, double **ud,
    double **x, double **u,
    struct ocp_qp_SOLVERNAME_args *args, double *work);
```

where `SOLVERNAME` is the name of the specific solver.

`N` [input] is the horizon length.

`nx` [input] is the vector of the state sizes n_u at the different stages, such that `nx[n]` is the state size at stage `n`.

nu [input] is the vector of the input sizes n_x at the different stages, such that `nu[n]` is the input size at stage `n`.

nb [input] is the vector of the bound sizes n_b at the different stages, such that `nb[n]` is the bound size at stage `n`. The value of `nb[n]` is smaller or equal to `nx[n]+nu[n]`.

ng [input] is the vector of the general polytopic constraint sizes n_g at the different stages, such that `ng[n]` is the general polytopic constraint size at stage `n`.

A [input] is the vector of size N of the pointers to the first element of the matrices A_n, such that `A[n]` is the pointer to the first element of the matrix A_n, and `A[n][0]` is the first element of the matrix A_n. The matrix referenced by the pointer `A[n]` is stored in column-major (or Fortran-like) order, in a vector of `nx[n+1]`$\times$`nx[n]` double-precision floating-point numbers.

B [input] is the vector of size N of the pointers to the first element of the matrices B_n, such that `B[n]` is the pointer to the first element of the matrix B_n, and `B[n][0]` is the first element of the matrix B_n. The matrix referenced by the pointer `B[n]` is stored in column-major (or Fortran-like) order, in a vector of `nx[n+1]`$\times$`nu[n]` double-precision floating-point numbers.

b [input] is the vector of size N of the pointers to the first element of the vectors b_n, such that `b[n]` is the pointer to the first element of the vector b_n, and `b[n][0]` is the first element of the vector b_n. The vector referenced by the pointer `b[n]` is stored in a vector of `nx[n+1]`$\times$`1` double-precision floating-point numbers.

Q [input] is the vector of size $N+1$ of the pointers to the first element of the matrices Q_n, such that `Q[n]` is the pointer to the first element of the matrix Q_n, and `Q[n][0]` is the first element of the matrix Q_n. The matrix referenced by the pointer `Q[n]` is stored in column-major (or Fortran-like) order, in a vector of `nx[n]`$\times$`nx[n]` double-precision floating-point numbers.

S [input] is the vector of size N of the pointers to the first element of the matrices S_n, such that `S[n]` is the pointer to the first element of the matrix S_n, and `S[n][0]` is the first element of the matrix S_n. The matrix referenced by the pointer `S[n]` is stored in column-major (or Fortran-like) order, in a vector of `nu[n]`$\times$`nx[n]` double-precision floating-point numbers.

R [input] is the vector of size N of the pointers to the first element of the matrices R_n, such that `R[n]` is the pointer to the first element of the matrix R_n, and `R[n][0]` is the first element of the matrix R_n. The matrix referenced by the pointer `R[n]` is stored in column-major (or Fortran-like) order, in a vector of `nu[n]`×`nu[n]` double-precision floating-point numbers.

q [input] is the vector of size $N+1$ of the pointers to the first element of the vectors q_n, such that `q[n]` is the pointer to the first element of the vector q_n, and `q[n][0]` is the first element of the vector q_n. The vector referenced by the pointer `q[n]` is stored in a vector of `nx[n]`×`1` double-precision floating-point numbers.

r [input] is the vector of size N of the pointers to the first element of the vectors r_n, such that `r[n]` is the pointer to the first element of the vector r_n, and `r[n][0]` is the first element of the vector r_n. The vector referenced by the pointer `r[n]` is stored in a vector of `nu[n]`×`1` double-precision floating-point numbers.

idxb [input] is the vector of size $N+1$ of the pointers to the first element of the integer vectors $idxb_n$ describing the indexes of the corresponding upper and lower bounds in `lb` and `ub`, such that `idxb[n]` is the pointer to the index of the first bound at stage n, and `idxb[n][0]` is index of the first bound at stage n. The indexes in `idxb[n]` correspond to the position of the constrained components in the variables vector $\begin{bmatrix} u_n \\ x_n \end{bmatrix}$: therefore a bound on the first input component has index `0`, a bound on the last input component has index `nu[n]-1`, a bound on the first state component has index `nu[n]` and a bound on the last state component has index `nu[n]+nx[n]-1`. The vector referenced by the pointer `idxb[n]` is stored in a vector of `nb[n]`×`1` integer numbers.

lb [input] is the vector of size $N+1$ of the pointers to the first element of the vectors $\begin{bmatrix} \underline{u}_n \\ \underline{x}_n \end{bmatrix}$, such that `lb[n]` is the pointer to the first element of the vector $\begin{bmatrix} \underline{u}_n \\ \underline{x}_n \end{bmatrix}$, and `lb[n][0]` is the first element of the vector $\begin{bmatrix} \underline{u}_n \\ \underline{x}_n \end{bmatrix}$. The vector referenced by the pointer `lb[n]` is stored in a vector of `nb[n]`×`1` double-precision floating-point numbers.

ub [input] is the vector of size $N+1$ of the pointers to the first element of the vectors $\begin{bmatrix} \overline{u}_n \\ \overline{x}_n \end{bmatrix}$, such that `ub[n]` is the pointer to the first element of the vector $\begin{bmatrix} \overline{u}_n \\ \overline{x}_n \end{bmatrix}$, and `ub[n][0]` is the first element of the vector $\begin{bmatrix} \overline{u}_n \\ \overline{x}_n \end{bmatrix}$. The vector referenced by the pointer `ub[n]` is stored in a vector of `nb[n]`×`1` double-precision floating-point numbers.

C [input] is the vector of size $N+1$ of the pointers to the first element of the matrices C_n, such that `C[n]` is the pointer to the first element of the matrix C_n, and `C[n][0]` is the first element of the matrix C_n. The matrix referenced by the pointer `C[n]` is stored in column-major (or Fortran-like) order, in a vector of `ng[n]`×`nx[n]` double-precision floating-point numbers.

D [input] is the vector of size N of the pointers to the first element of the matrices D_n, such that `D[n]` is the pointer to the first element of the matrix D_n, and `D[n][0]` is the first element of the matrix D_n. The matrix referenced by the pointer `D[n]` is stored in column-major (or Fortran-like) order, in a vector of `ng[n]`×`nu[n]` double-precision floating-point numbers.

ld [input] is the vector of size $N+1$ of the pointers to the first element of the vectors $\underline{d}_n$, such that `ld[n]` is the pointer to the first element of the vector $\underline{d}_n$, and `ld[n][0]` is the first element of the vector $\underline{d}_n$. The vector referenced by the pointer `ld[n]` is stored in a vector of `ng[n]`×`1` double-precision floating-point numbers.

ud [input] is the vector of size $N+1$ of the pointers to the first element of the vectors $\overline{d}_n$, such that `ud[n]` is the pointer to the first element of the vector $\overline{d}_n$, and `ud[n][0]` is the first element of the vector $\overline{d}_n$. The vector referenced by the pointer `ud[n]` is stored in a vector of `ng[n]`×`1` double-precision floating-point numbers.

x [output] is the vector of size $N+1$ of the pointers to the first element of the vectors x_n, such that `x[n]` is the pointer to the first element of the vector x_n, and `x[n][0]` is the first element of the vector x_n. The vector referenced by the pointer `x[n]` is stored in a vector of `nx[n]`×`1` double-precision floating-point numbers.

u [output] is the vector of size $N+1$ of the pointers to the first element of the vectors u_n, such that `u[n]` is the pointer to the first element of the vector u_n, and `u[n][0]` is the first element of the vector u_n. The vector referenced by the pointer `u[n]` is stored in a vector of `nu[n]`×`1` double-precision floating-point numbers.

args [input] is the pointer to a structure of type `ocp_qp_SOLVERNAME_args` that defines the arguments (as e.g. maximum number of iterations, minimum step size, ...) passed to the specific solver.

work [workspace] is the pointer to the working space used by the specific solver. The working space size (in doubles) is returned by a call to the function `ocp_qp_SOLVERNAME_workspace_double(int N, int *nx, int *nu, int *nb, int *ng, struct ocp_qp_solver_args *args)`.

Furthermore, the function returns an int, that is defined in the following enum (TODO change the names to something better!!!):

ACADOS_SUCCESS Solution successfully found.

ACADOS_MAXITER Maximum number of iterations reached.

ACADOS_MINSTEP Minumum step size reached (in IPs, probably unfeasible problem).

1.1.1 Examples

MPC problem

In the MPC problem, the initial state is fixed. This is modelled by choosinig `nx[0]=0`, i.e. not considering the initial state as an optimization variable. As a consequence, e.g. the matrix `A[0]` has size `nx[1]`×`0`, the matrix `Q[0]` has size `0`×`0`, and the vector `q[0]` has size `0`×`1`. The information about the known value of x_0 and the matrix A_0 are used to compute the value of the vector `b[0]`, that is initialized to $b_0 + A_0 \cdot x_0$.