An N^2 and n_x^2 Condensing Method for Solution of Linear-Quadratic Control Problems

Gianluca Frison

Technical Univeristy of Denmark

Embedded Quadratic Programming Workshop,
Freiburg, 19 March 2014

Linear quadratic control problem

Linear Quadratic Control Problem:

$$\begin{aligned}\min_{x,u}\quad & \sum_{n=0}^{N-1}\left(\frac{1}{2}\begin{bmatrix}x_n' & u_n'\end{bmatrix}\begin{bmatrix}Q_n & S_n' \\ S_n & R_n\end{bmatrix}\begin{bmatrix}x_n \\ u_n\end{bmatrix}+\begin{bmatrix}q_n' & s_n'\end{bmatrix}\begin{bmatrix}x_n \\ u_n\end{bmatrix}+\rho_n\right)+ \\ & +\frac{1}{2}x_N'Px_N+px_N+\rho_N \\ s.t.\quad & x_{n+1}=A_nx_n+B_nu_n+b_n \\ & x_0=\bar{x}_0\end{aligned}$$

General formulation:

- quadratic & linear cost function
- affine dynamic
- time variant matrices

Subproblem in IP methods

Linear Quadratic Control Problem:

$$
\begin{aligned}
\min_{x,u} \quad & \sum_{n=0}^{N-1} \left(\frac{1}{2} \begin{bmatrix} x_n' & u_n' \end{bmatrix} \begin{bmatrix} Q_n & S_n' \\ S_n & R_n \end{bmatrix} \begin{bmatrix} x_n \\ u_n \end{bmatrix} + \begin{bmatrix} q_n' & s_n' \end{bmatrix} \begin{bmatrix} x_n \\ u_n \end{bmatrix} + \rho_n \right) + \\
& + \frac{1}{2} x_N' P x_N + p x_N + \rho_N \\
s.t. \quad & x_{n+1} = A_n x_n + B_n u_n + b_n \\
& x_0 = \bar{x}_0
\end{aligned}
$$

Problem size:

- n_x states number
- n_u inputs number
- N horizon length

KKT system

- the LQ control problem is an equality constrained QP

$$\begin{aligned} \min_{\theta} \quad & \tfrac{1}{2}\theta' H\theta + h'\theta \\ s.t. \quad & G\theta = g \end{aligned}$$

- KKT necessary (and sufficient with mild assumptions) conditions

$$\begin{bmatrix} H & -G' \\ -G & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \pi \end{bmatrix} = - \begin{bmatrix} h \\ g \end{bmatrix}$$

- KKT matrix symmetric, sparse and structured, of size $N(2n_x + n_u)$

Structure of the KKT system

- symmetric and indefinite: can be solved using LDL factorization in $\mathcal{O}(N^3(2n_x + n_u)^3)$ flops (naive approach)

$$
\left[\begin{array}{ccc|ccc|ccc}
R_0 & & & & & & B_0' & & \\
& R_1 & & S_1 & & & & B_1' & \\
& & R_2 & & S_2 & & & & B_2' \\
\hline
& S_1' & & Q_1 & & & -I & A_1' & \\
& & S_2' & & Q_2 & & & -I & A_2' \\
& & & & & Q_3 & & & -I \\
\hline
B_0 & & & -I & & & & & \\
& B_1 & & A_1 & -I & & & & \\
& & B_2 & & A_2 & -I & & &
\end{array}\right]
\left[\begin{array}{c}
u_0 \\ u_1 \\ u_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ \hline \pi_1 \\ \pi_2 \\ \pi_3
\end{array}\right]
=
\left[\begin{array}{c}
-\tilde{s}_0 \\ -s_1 \\ -s_2 \\ \hline -q_1 \\ -q_2 \\ -q_3 \\ \hline -\tilde{b}_0 \\ -b_1 \\ -b_2
\end{array}\right]
$$

Solution of the KKT system - Riccati recursion

The Riccati recursion is a factorization of the KKT matrix rewritten in the form [Rao, Wright and Rawlings (1998)]

$$
\begin{bmatrix}
R_0 & B_0' & & & & & & & \\
B_0 & & -I & & & & & & \\
 & -I & Q_1 & S_1' & A_1' & & & & \\
 & & S_1 & R_1 & B_1' & & & & \\
 & & A_1 & B_1 & & -I & & & \\
 & & & & -I & Q_2 & S_2' & A_2' & \\
 & & & & & S_1 & R_1 & B_1' & \\
 & & & & & A_2 & B_2 & & -I \\
 & & & & & & & -I & P
\end{bmatrix}
\begin{bmatrix}
u_0 \\ \lambda_1 \\ x_1 \\ u_1 \\ \lambda_2 \\ x_2 \\ u_2 \\ \lambda_3 \\ x_3
\end{bmatrix}
= -
\begin{bmatrix}
\tilde{s}_0 \\ \tilde{b}_0 \\ q_1 \\ s_1 \\ b_1 \\ q_2 \\ s_2 \\ b_2 \\ p
\end{bmatrix}
$$

- non-condensed approach exploiting the KKT matrix structure
- cost $\mathcal{O}(N(n_x + n_u)^3)$

Solution of the KKT system - Condensing

- state elimination

$$\bar{x} = \Gamma \bar{u} + \bar{A}^{-1} \bar{b}$$

where

$$\Gamma = \begin{bmatrix} I & & \\ -A_1 & I & \\ & -A_2 & I \end{bmatrix}^{-1} \begin{bmatrix} B_0 & & \\ & B_1 & \\ & & B_2 \end{bmatrix} = \begin{bmatrix} B_0 & & \\ A_1 B_0 & B_1 & \\ A_2 A_1 B_0 & A_2 B_1 & B_2 \end{bmatrix}$$

- only inputs as optimization variables

$$H \bar{u} = f$$

where

$$H = \bar{R} + \Gamma' \bar{S}' + \bar{S} \Gamma + \Gamma' \bar{Q} \Gamma$$

Solution of the KKT system - Condensing

- the large, sparse and structured KKT system is rewritten into a small and dense system of linear equations
- this system has size Nn_u and it is positive definite
- it is traditionally solved using Cholesky factorization and forward and backward substitution: the cost is $\mathcal{O}(N^3 u_u^3)$ flops
- is there still structure left in the small, dense condensed system? yes

Cholesky factorization

- 2×2 block version of the algorithm

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = U'U = \begin{bmatrix} U'_{11} & \\ U'_{12} & U_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & U_{22} \end{bmatrix} =$$
$$= \begin{bmatrix} U'_{11}U_{11} & U'_{11}U_{12} \\ U'_{12}U_{11} & U'_{22}U_{22} + U'_{12}U_{12} \end{bmatrix}$$

- We can apply the procedure recursively:
  1. factorize H_{11} to get U_{11}
  2. solve $U_{11}^{-T} H_{12}$ to get U_{12}
  3. correct H_{22} to get $H_{22} - U'_{12}U_{12} = U'_{22}U_{22} \doteq \tilde{H}_{22}$
  4. repeat recursively on $\tilde{H}_{22}$
- Cost:
$\sum_{i=1}^{n} 1 + (i-1) + \frac{2i(i-1)}{2} = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{1}{3}n^3$
- Notice that the factorization starts from the top-left block

Structure of the condensed matrix

- for the moment let us assume that $S_n = 0$ (only for clarity of presentation)
- for $N = 3$, the condensed matrix looks already pretty complicated

$$\begin{bmatrix} R_0 + B_0' Q_1 B_0 + B_0' A_1' Q_2 A_1 B_0 + B_0' A_1' A_2' P_3 A_2 A_1 B_0 & B_0' A_1' Q_2 B_1 + B_0' A_1' A_2' P_3 A_2 B_1 & B_0' A_1' A_2' P_3 B_2 \\ B_1' Q_2 A_1 B_0 + B_1' A_2' P_3 A_2 A_1 B_0 & R_1 + B_1' Q_2 B_1 + B_1' A_2' P_3 A_2 B_1 & B_1' A_2' P_3 B_2 \\ B_2' P_3 A_2 A_1 B_0 & B_2' P_3 A_2 B_1 & R_2 + B_2' P_3 B_2 \end{bmatrix}$$

- complex structure at the top-left corner
- simple structure at the bottom-right corner
- what if we permute the matrix?

Factorization of the permuted condensed matrix

- Let us reverse all columns and rows, and apply Cholesky factorization (for $N = 2$)

$$\begin{bmatrix} R_1 + B_1' P_2 B_1 & B_1' P_2 A_1 B_0 \\ B_0' A_1 P_3 B_1 & R_0 + B_0' Q_1 B_0 + B_0' A_1' P_2 A_1 B_0 \end{bmatrix}$$

- factorize $R_1 + B_1' P_2 B_1 = U_{11}' U_{11}$
- solve $U_{12} = U_{11}^{-T} (B_1' P_2 A_1 B_0)$
- correct $R_0 + B_0' Q_1 B_0 + B_0' A_1' P_2 A_1 B_0 - U_{12}' U_{12} \doteq R_0 + B_0' P_1 B_0$

where $P_1 = Q_1 + A_1' P_2 A_1 - A_1' P_2 B_1 (R_1 + B_1' P_2 B_1)^{-1} B_1' P_2 A_1$

- that is the classical Riccati recusion

Factorization of the permuted condensed matrix

[D. Axehill, M. Morari (2012)]

- Riccati recursion can be used to compute the factorization of the dense Hessian matrix
- the factorized system is solved using standard backward and forward substitutions
- Riccati recursion for the computation of the matrices P_n: cost $\mathcal{O}(N(n_x + n_u)^3)$
- construction of the Cholesky factor of H: $\mathcal{O}(N^2)$
- no $\mathcal{O}(N^3)$ operations, but the overall algorithm is always slower that Riccati recursion
- can we get an algorithm with better complexity? yes

Structure exposed

For $N = 3$, we can write the permuted matrix as

$$\begin{bmatrix} R_2 + B_2' P_3 B_2 & (B_2' P_3 A_2) B_1 & (B_2' P_3 A_2) A_1 B_0 \\ * & R_1 + B_1' Q_2 B_1 + B_1' A_2' P_3 A_2 B_1 & (B_1' Q_2 A_1 + B_1' A_2' P_3 A_2 A_1) B_0 \\ * & * & R_0 + B_0' Q_1 B_0 + B_0' A_1' Q_2 A_1 B_0 + B_0' A_1' A_2' P_3 A_2 A_1 B_0 \end{bmatrix} =$$

$$= \begin{bmatrix} D_2 & M_2 B_1 & M_2 A_1 B_0 \\ * & D_1 & M_1 B_0 \\ * & * & D_0 \end{bmatrix}$$

- dense matrix, but now structure is exposed
- is Cholesky factorization preserving this structure? yes

Structure exposed - factorization - 1st row

- factorization

$$\begin{bmatrix} U_2 & M_2 B_1 & M_2 A_1 B_0 \\ * & D_1 & M_1 B_0 \\ * & * & D_0 \end{bmatrix}$$

- solution (key idea: update of one single matrix $\Rightarrow$ no $\mathcal{O}(N^2)$ terms))

$$\begin{bmatrix} U_2 & U_2^{-T} M_2 B_1 & U_2^{-T} M_2 A_1 B_0 \\ * & D_1 & M_1 B_0 \\ * & * & D_0 \end{bmatrix}$$

- correction (key idea: the correction the block H_{22} is equivalent to the correction of the matrix Q_2 $\Rightarrow$ no $\mathcal{O}(N^3)$ terms)

$$\begin{bmatrix} U_2 & L_2 B_1 & L_2 A_1 B_0 \\ * & \tilde{D}_1 & \tilde{M}_1 B_0 \\ * & * & \tilde{D}_0 \end{bmatrix}$$

  - $\tilde{D}_1 = D_1 - B_1' L_2' L_2 B_1 = R_1 + B_1'(Q_2 - L_2' L_2) B_1 + B_1' A_2' P_3 A_2 B_1$
  - $\tilde{M}_1 = M_1 - B_1' L_2' L_2 A_1 = B_1'(Q_2 - L_2' L_2) A_1 + B_1' A_2' P_3 A_2 A_1$
  - $\tilde{D}_0 = D_0 - B_0' A_1' L_2' L_2 A_1 B_0 =$
    $R_0 + B_0' Q_1 B_0 + B_0' A_1'(Q_2 - L_2' L_2) A_1 B_0 + B_0' A_1' A_2' P_3 A_2 A_1 B_0$

Structure exposed - factorization - 2nd row

- factorization

$$\begin{bmatrix} U_2 & M_2 B_1 & M_2 A_1 B_0 \\ * & U_1 & M_1 B_0 \\ * & * & D_0 \end{bmatrix}$$

- solution

$$\begin{bmatrix} U_2 & L_2 B_1 & L_2 A_1 B_0 \\ * & U_1 & U_1^{-T} \tilde{M}_1 B_0 \\ * & * & \tilde{D}_0 \end{bmatrix}$$

- correction

$$\begin{bmatrix} U_2 & L_2 B_1 & L_2 A_1 B_0 \\ * & U_1 & L_1 B_0 \\ * & * & \bar{D}_0 \end{bmatrix}$$

- $\bar{D}_0 = \tilde{D}_0 - B_0' L_1' L_1 B_0 = R_0 + B_0'(Q_1 - L_1' L_1) B_0 + B_0' A_1' (Q_2 - L_2' L_2) A_1 B_0 + B_0' A_1' A_2' P_3 A_2 A_1 B_0$

Structure exposed - factorization - 3rd row

- factorization only

$$\hat{\mathcal{U}} = \begin{bmatrix} U_2 & L_2 B_1 & L_2 A_1 B_0 \\ & U_1 & L_1 B_0 \\ & & U_0 \end{bmatrix}$$

- key idea: the matrix $\hat{\mathcal{U}}$ is build and factorized on-the-fly, once the corrected Q_n matrices are computed
- can this structure be exploited also to solve the factorized system

$$\hat{\mathcal{U}}' \hat{\mathcal{U}} \hat{u} = -\hat{f}$$

using forward and backward substitution? yes

Structure exposed - system solution

- forward substitution

$$\begin{bmatrix} v_2 \\ v_1 \\ v_0 \end{bmatrix} = - \begin{bmatrix} U_2^{-T}(g_2) \\ U_1^{-T}(g_1 + B_1' L_2' v_2) \\ U_0^{-T}(g_0 + B_0' A_1' L_2' v_2 + B_0' L_1' y_1) \end{bmatrix}$$

- backward substitution

$$\begin{bmatrix} u_2 \\ u_1 \\ u_0 \end{bmatrix} = \begin{bmatrix} U_2^{-1}(v_2 - L_2 B_1 u_1 - L_2 A_1 B_0) \\ U_1^{-1}(v_1 - L_1 B_0 u_0) \\ U_0^{-1}(v_0) \end{bmatrix}$$

- key idea: we do not even need to explicitly build $\hat{\mathcal{U}}$, we just need to compute the matrices U_n and L_n (and thus in turn D_n and M_n)

Structure exposed - cost

- the cost of the factorization is then linear in N

$$\frac{1}{3}Nn_u^3 + (N-1)n_x n_u^2 + (N-1)n_x^2 n_u$$

  plus the cost to build D_n and M_n
- two approaches to build D_n and M_n
  1. avoid $\mathcal{O}(N^2)$ operations, at the cost of higher complexity in n_x
  2. avoid $\mathcal{O}(n_x^3)$ operations, at the cost of higher complexity in N
- the most efficient approach depends on the problem size

Build the matrix - 1st approach

Riccati-like solver: use a recursion to keep a constant number of operations per stage

$$D_n = R_n + (B_n' P_{n+1}) B_n$$
$$M_n = S_n + (B_n' P_{n+1}) A_n$$

where

$$P_{n+1} = Q_{n+1}^* + A_n' P_{n+1} A_n = (Q_{n+1} - L_n' L_n) + A_n' P_{n+1} A_n$$

- the computation of $A_n' P_{n+1} A_n$ is cubic in n_x
- total cost (build+factorize): $N(\frac{7}{3} n_x^3 + 4 n_x^2 n_u + 2 n_x n_u^2 + \frac{1}{3} n_u^3)$

Build the matrix - 2nd approach

Pure condensing solver: always multiply matrices of size $n_x \times n_x$ to matrices of size $n_x \times n_u$

$$\hat{D} = \hat{R} + \hat{B}' \cdot \mathrm{diag}\big(\hat{A}^{-T}(\hat{Q}^* \cdot \hat{\Gamma})\big)$$
$$\hat{M} = \hat{S} + \big(\mathrm{diag}\big(\hat{A}^{-T}(\hat{Q}^* \cdot \hat{\Gamma})\big)\big)' \cdot \hat{\mathcal{A}}$$

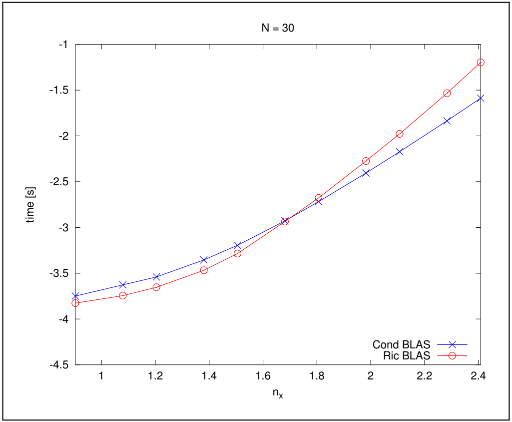
where

$$\hat{\mathcal{A}} = \begin{bmatrix} 0 & A_2 & \\ & 0 & A_1 \\ & & 0 \end{bmatrix}$$

- in an IP method, $\hat{\Gamma} = \hat{A}^{-1} \cdot \hat{B}$ can be computed off-line
- $\hat{Q}^* \cdot \hat{\Gamma}$ and $\hat{A}^{-T}(\hat{Q}^*\hat{\Gamma})$ cost $N^2 n_x^2 n_u$
- total cost (build+factorize): $2N^2 n_x^2 n_u + 3N n_x n_u^2 + \frac{1}{3} N n_u^3$

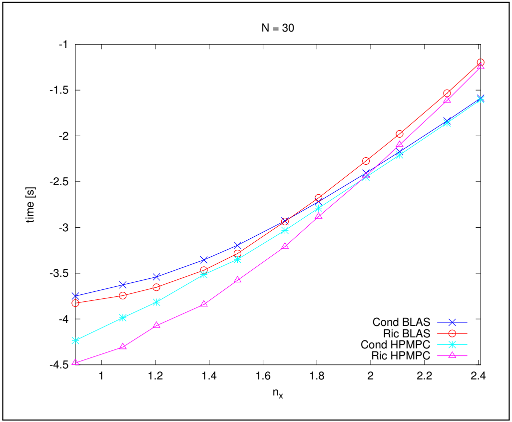
Numerical results # 1

- n_x varying
- Riccati $\mathcal{O}(n_x^3)$ vs Condensing $\mathcal{O}(n_x^2)$
- OpenBLAS



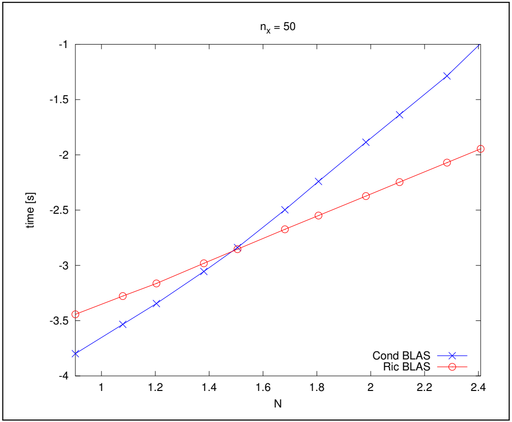
Numerical results # 1

- n_x varying
- Riccati $\mathcal{O}(n_x^3)$ vs Condensing $\mathcal{O}(n_x^2)$
- OpenBLAS vs HPMPC



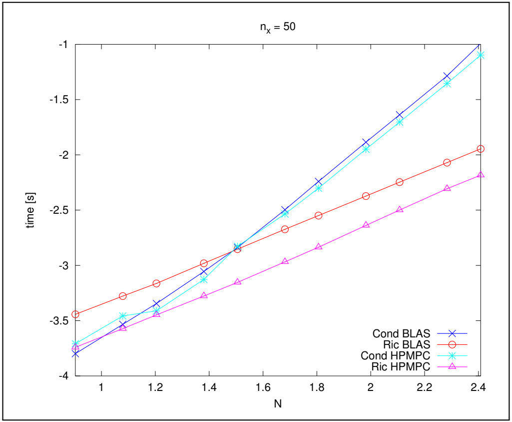
Numerical results # 2

- N varying
- Riccati $\mathcal{O}(N)$ vs Condensing $\mathcal{O}(N^2)$
- OpenBLAS



Numerical results # 2

- N varying
- Riccati $\mathcal{O}(N)$ vs Condensing $\mathcal{O}(N^2)$
- OpenBLAS vs HPMPC



Conclusion

- structure-exploiting factorization of the condensed Hessian
- factorization cost is linear in N, plus the cost to build D_n and M_n
- 1st approach: Riccati-like solver, cost linear in N and cubic in n_x
- 2nd approach: pure condensing solver, cost quadratic in N and quadratic in n_x

Questions?