A sparse model predictive control formulation for walking motion generation

Dimitar Dimitrov¹, Alexander Sherikov¹, Pierre-Brice Wieber²

¹Örebro University, Sweden
²INRIA, Grenoble, France

September 10, 2011
Scenario
- humanoid robot walks on a flat surface
- assumption
- the system could be subject to external disturbances
- higher level planner generates reference footsteps

Objective
follow reference footsteps while preserving the “stability” of the system

Required
a scheme for online trajectory following and stabilization
Trajectory following + stabilization ≜ walking motion generation

How to approach the problem *(in under 2 ms)*

- using predefining motion primitives - **not possible** in the presence of disturbances
- making local decisions considering the full dynamical morel - **not reliable**
- “look-ahead” schemes - increasingly popular but computationally demanding. In particular, using full system dynamics - **not feasible**.

One possible “solution”

- use approximate dynamical model (preferably linear)
- compensate the approximation by applying a preview type of controller with (possibly) fast sampling rate
We use

- **model**: linearized 3D inverted pendulum - surprisingly accurate approximation (under certain assumptions)
- **preview controller**: Linear Quadratic Regulator (LQR) with explicit constraints $\triangleq$ Linear Model Predictive Control (LMPC)
- **stability criterion**: $\text{ZMP} \in \text{support polygon}$

**Explicit constraints** - address the stabilization sub-task

**Figure**: A typical result (fixed feet). Red squares - ZMP, blue line - CoM. Double support constraints are not displayed.
The paper deals with efficient implementation

Linear dynamical system

\[ x_{k+1} = Ax_k + Bu_k \]
\[ x_0 \] is a known initial state
\[ k = 0, \ldots, N - 1 \]

Quadratic objective function (to minimize)

\[ J(v_x, v_u) = v_x^T H_x v_x + v_u^T H_u v_u \]

number of variables: \( N_x + N_u \)
\[ v_x = (x_1, \ldots, x_N), \quad v_u = (u_0, \ldots, u_{N-1}) \]

The “standard approach”

Reformulate problem using **minimal number of variables** \( N_u \). Or in other words, eliminate equality constraints due to the dynamics of the system.

\[ v_x = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} w \]
\[ x_0 + \begin{bmatrix} B \\ AB \\ \vdots \\ A^{N-1}B \end{bmatrix} \begin{bmatrix} 0 \\ B \\ \vdots \\ A^{N-2}B \end{bmatrix} v_u \]

\[ J(v_u) = (wx_0 + Wv_u)^T H_x (wx_0 + Wv_u) + v_u^T H_u v_u \]
Some drawbacks of eliminating the equality constraints

\[
\begin{align*}
\text{minimize} & \quad J(\mathbf{v}_u) = (w\mathbf{x}_0 + W\mathbf{v}_u)^T H_x (w\mathbf{x}_0 + W\mathbf{v}_u) + \mathbf{v}_u^T H_u \mathbf{v}_u \\
& = \mathbf{v}_u^T \left( W^T H_x W + H_u \right) \mathbf{v}_u + \ldots
\end{align*}
\]

- the new Hessian matrix \( H \) is in general dense - the structure of the problem is lost
- forming the product \( W^T H_x W \) is expensive and usually has to be performed offline
- the computational cost per iteration is \( O(N^3) \) (for interior-point methods) and \( O(N^2) \) (for active-set methods)

In the context of our application the matrices \( H_x \) and \( W \) contain information about
- discretization of the preview window
- height of the CoM

Point 2 above implies that both should be constant.
As opposed to \textit{condensing} the problem, one can solve directly

\[
\begin{align*}
\text{minimize} & \quad \mathbf{v}_x, \mathbf{v}_u \quad \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_u \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_x & 0 \\ 0 & \mathbf{H}_u \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_u \end{bmatrix} \\
\text{subject to} & \quad \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = 0, \ldots, N - 1 \\
& \quad \mathbf{x}_0 \text{ is a known initial state.}
\end{align*}
\]

This is a larger but more structured quadratic program (QP)

\begin{itemize}
  \item solution can be obtained at a cost of $O(N)$ per iteration (\textit{e.g.}, by using Riccati recursion)
  \item there is no need to pre-compute the objective function offline
  \item CoM height and discretization sampling can be altered online (at practically no additional computational cost)
\end{itemize}
How about inequality constraints?

We perform a change of variable that leads to a simplified formulation

**Standard approach**

- **control input**: jerk of CoM
- **output**: position of ZMP

\[ \text{jerk of CoM} \rightarrow \text{system dynamics} \rightarrow \text{position of ZMP} \]

\[ \Rightarrow \text{The system dynamics appears in the constraints for the ZMP} \]

We use the ZMP directly as a decision variable

\[ \text{minimize usual stuff} \]

\[ \text{ZMP} \]

subject to \( \text{ZMP} \in \text{support polygon} \leftarrow \text{pure geometry} \)

In this way we can derive a formulation with

- simple bounds
- diagonal Hessian matrix
Numerical results (active set method)

Preview window 1.5 s
- \( N = 75 \)
- \( T = 20 \) ms

Dense formulation (off-the-shelf solver QL)
- \# variables: 150
- \# eq. constraints: 0
- \# simple bounds: 150

Sparse formulation (custom-made solver)
- \# variables: 600
- \# eq. constraints: 450
- \# simple bounds: 150

C++ implementation available for download at https://github.com/asherikov/smpc_solver