PolyMPC: An efficient Tool for Embedded Model Predictive Control for Fast Mechatronic Systems

Petr Listov, Michael Spieler, Colin Jones
École polytechnique fédérale de Lausanne

ABSTRACT

Efficient solvers for real-time Nonlinear Model Predictive Control (NMPC) are needed to run on embedded hardware with highly constrained computational resources. Most existing solvers are based on second-order methods which are prohibitively expensive for some applications. This project explores a Sequential Quadratic Programming (SQP) strategy using an ADMM based Quadratic Program (QP) solver. It is implemented as a generic nonlinear solver in form of a header-only module, which integrates into PolyMPC, an open-source C++ library for real-time NMPC. A pseudospectral collocation based approximation method is used to efficiently solve the QP, while forward mode automatic differentiation simplifies the problem construction. We leverage the flexibility of templated C++ with the Eigen linear algebra library to solve QCPs without relying on dynamic memory allocation. The implementation is suitable to run on a microcontroller with Floating Point Unit (FPU), which was tested on a path following problem of a two-line soft-wing kite.

KEY FEATURES AND PRIMER

Efficiency: compile-time polymorphism and generic programming in C++ are used to avoid expensive calling of virtual methods.

Modularity: the software does not use any custom defined modelling language, but rather relies on popular dynamic optimization and linear algebra frameworks. This allows one to use implemented tracking and path-following predictive controllers, as well as utilize the building blocks of the algorithms such as the polynomial interpolation, collocation of differential equations and quadrature rule approximations of integrals.

Usability: PolyMPC does not use code generation, algorithms are implemented in a compact and readable way.

Automatic Differentiation: CasADi (forward and backward modes) and Eigen AutoDiff module (only forward mode, can work on microcontrollers).

f(x₁,...,xₙ) = \sum_{i=1}^{n} [a - x_i]^2 + b(x_{i+1} - x_i)^2

Deployment: in-house SQP and QP solvers, no third party libraries (except Eigen) and dynamic memory allocation for embedded applications.

Memory Optimization: support for direct and indirect linear algebra solvers, in-place LDLT decomposition, cheap damped BFGS Hessian updates, support single and double precision.

SIMULATION RESULTS

Kite model equations:
\[
\begin{bmatrix}
L & 0 \\
0 & L \cos \theta
\end{bmatrix} \dot{\theta} = R_{xx} [1 0 \ -E] R_{xx} R_{xx} v_w - R_{xx} [E_2]
\]

Where

\[
R_{xx} = \begin{bmatrix}
-\sin \theta \cos \phi \\
-\sin \theta \sin \phi \\
\cos \phi
\end{bmatrix}
\]

Path equations:

\[
\theta(\gamma) = h + a \sin(2\gamma)
\]

\[
\psi(\gamma) = 4a \cos(2\gamma)
\]

The simulation was run at 100 Hz, and the controller node was set to run at 50Hz with the following parameters: number of collocation points- 12; number of subintervals- 3, prediction horizon- 1.5 seconds.

TESTING ON EMBEDDED PLATFORMS

<table>
<thead>
<tr>
<th>Name</th>
<th>Odroid XU4</th>
<th>Nexus-F767ZI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Samsung Exynos5422</td>
<td>STM32F767ZI</td>
</tr>
<tr>
<td>CPU</td>
<td>8x ARM Cortex M4-A7</td>
<td>ARM Cortex M7</td>
</tr>
<tr>
<td>Architecture</td>
<td>ARMv7A (32bit)</td>
<td>ARMv7E-M (32bit)</td>
</tr>
<tr>
<td>Acceleration</td>
<td>FPV, NEON SIMD, DSP</td>
<td>FPV (DP-16++)</td>
</tr>
<tr>
<td>RAM</td>
<td>2GHz</td>
<td>216MHz</td>
</tr>
<tr>
<td>Storage</td>
<td>2GB LPDDR3</td>
<td>512KB SRAM</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu 18.04</td>
<td>bare metal</td>
</tr>
<tr>
<td>Dimensions</td>
<td>88mm x 55mm</td>
<td>133mm x 70mm</td>
</tr>
<tr>
<td>Power consumption</td>
<td>10W - 20W</td>
<td>&lt;1.5W</td>
</tr>
</tbody>
</table>

REFERENCES