Multi-Parametric Toolbox 3.0

Martin Herceg
Manfred Morari

Michal Kvasnica

Colin N. Jones

ETH Zürich
STU Bratislava
EPFL Lausanne
What is MPT?

- Matlab toolbox for application of explicit MPC
  - high-speed implementation of MPC in real-time
- Approach
  - **offline**: solve optimal control problem parametrically
  - **online**: evaluate the resulting PWA feedback

Over 30,000 downloads in 10 years!
Core Features of Version 3.0

- New engine for parametric optimization
  - new parametric and non-parametric solvers

- Extended geometric library
  - convex sets and function over sets

- More flexible MPC design
  - modular structure, object oriented

- Novel algorithms for reduction of complexity
  - separation, clipping, PWA fitting, ...
Core Numerical Engines

- **New parametric solver – PLCP**
  - relies on solving linear-complementarity problems (LCP) by approach of Jones, Morari, CDC’06
  - features lexicographic perturbations to improve robustness

- **New nonparametric solver – LCP**
  - implements lexicographic Lemke’s algorithm

- **Interfaces to state-of-the-art solvers**
  - CPLEX, GUROBI, NAG, CDD, GLPK, QPOASES, QPspline

\[
\begin{align*}
\min f^T x \\
Ax &\leq b
\end{align*}
\]

problem = Opt('f', f, 'A', A, 'b', b)
solution = problem.solve
Parametric Optimization

- Single solver for parametric linear and quadratic problems

\[
\begin{align*}
\min & \frac{1}{2} u^T H u + c^T u \\
\text{subject to} & \quad A u \leq b + E \xi, \quad u \geq 0
\end{align*}
\]

\[
Hu + c + A^T \lambda - \nu = 0
\]
\[
Au \leq b + E \xi, \quad u \geq 0
\]
\[
\lambda^T (Au - b - E \xi) = 0, \quad \nu^T u = 0
\]

LCP formulation

\[
\begin{align*}
\text{find } w, z \\
w - Mz &= q + G \xi \\
w &\geq 0, \quad z \geq 0 \\
w^T z &= 0
\end{align*}
\]
Geometric Library – Basic Sets

- Extended support for polyhedra
  - unbounded, lower-dimensional sets

```python
P = Polyhedron('A, A, 'b', b, 'Ae', Ae, 'be', be)
Q = Polyhedron('V', V, 'R', R)
```

- **unions of polyhedra** with certain properties

```python
U = PolyUnion('Set', P, 'Convex', 1, 'Overlaps', 0)
```
Geometric Library – Operations

- Supported geometric operations
  - Minkowski summations, Pontryagin differences, affine maps, projections, set-differences, convex hulls, ...

\[ S = P + Q \]

\[ T = U \setminus S \]

All algorithms built with MPT2 still work!
Geometric Library – Extensions

- **General convex sets**
  - import of YALMIP constraints as **YSet** objects

  ```
  x = sdpvar(2,1);
  box = ( [0;0.5] <= x <= [1; 2] );
  circle = ( 2*norm(x-1) <= 1 );
  S = YSet(x, box + circle);
  ```

- **Functions over sets**
  - represented compactly as **PolyUnion** objects

  \[
  f(x) = f_i(x) \text{ if } x \in R_i
  \]
New MPC Setup

- Basic **MPCController** object
  - represents constrained finite horizon optimal control problem

\[
\min \sum_{i=0}^{N} (\|Q x_k\| + \|R u_k\|)
\]

s.t.: \( x_{k+1} = f(x_k, u_k) \)
\( x \in X, u \in U \)

- Flexible interface for formulating control problems
  - based on modularized code
  - all controllers are derived from a common object

- **Support for constrained linear and hybrid models**
  - LTI models, PWA models, and MLD models
Versatility of LTI Models

- Autonomous system  \( x_{k+1} = Ax_k \)
  
  \[
  \text{model} = \text{LTISystem}('A', A)
  \]

- Affine autonomous system  \( x_{k+1} = Ax_k + f \)
  
  \[
  \text{model} = \text{LTISystem}('A', A, 'f', f)
  \]

- State update equation  \( x_{k+1} = Ax_k + Bu_k \)
  
  \[
  \text{model} = \text{LTISystem}('A', A, 'B', B)
  \]

- Output equation  \( x_{k+1} = Ax_k + Bu_k, \ y_k = Cx_k \)
  
  \[
  \text{model} = \text{LTISystem}('A', A, 'B', B, 'C', C)
  \]
Constraints

- **Constraints**
  - **lower/upper bounds** on signals

\[-1 \leq u \leq 1\]

```python
model.u.min = -1; model.u.max = 1;
```

- specific constraints can be added using the concept of filters (blocking constraints, rate constraints, logical constraints, soft constraints, ...)

\[x_N \in \Omega, \quad \Omega = \{x \mid Ax \leq b\}\]

```python
model.x.with('terminalSet')
model.x.terminalSet = Polyhedron(A, b)
```
Performance Specifications

- Penalties on system signals

\[ x^T Q x \]  

\[ \|Qx\|_1 \]  

\[ \|Qx\|_\infty \]

- additional penalties can be provided as filters, e.g. terminal penalties, slew-rate penalization, tracking of references, ...

MPT2 setups can be seamlessly imported!

\[
\text{model} = \text{mpt\_import}(\text{sysStruct},\text{probStruct})
\]
Generation of Explicit Solution

1. **Construct the online MPC controller object**
   
   ```
   ctrl = MPCController(model, N)
   u = ctrl.evaluate(x)
   ```

2. **Tune the controller** and close the loop
   
   ```
   loop = ClosedLoop(ctrl, model)
   data = loop.simulate(x0, Nsim)
   ```

3. **Export to the explicit form**
   
   ```
   expl_ctrl = ctrl.toExplicit()
   ```
Fine Tuning

- Tuning and refinement of MPC setups using YALMIP
  - export to YALMIP
    \[ Y = \text{ctrl.toYALMIP}() \]
  - adjust constraints and performance specification
  - construct back the online MPC object
    \[ \text{ctrl.fromYALMIP}(Y) \]

Arbitrary adjustments are possible!
Deployment of Explicit Controllers

- Export to low level programming language – code generation
  
  ```
  expl_ctrl.exportToC()
  ```

- Includes routines for **high speed evaluation**
  - consecutive search
  - binary search tree

- Test in Simulink and deploy on real-time platform
2D Example

- Oscillator example
  - CFTOC with horizon 5
  - cost function
    ```
    expl_ctrl.cost.fplot()
    ```
  - feedback law
    ```
    expl_ctrl.feedback.fplot()
    ```
  - partition
    ```
    expl_ctrl.partition.plot()
    ```
Summary

- **MPT 3.0**
  - novel parametric optimization engine
  - contains powerful library for computational geometry
  - flexible MPC synthesis
  - export to C code

- **Documentation**

- **Support**
  - Feedback is welcome!
  - Enquiries and suggestions mpt@control.ee.ethz.ch

http://control.ee.ethz.ch/~mpt/3