Multi-Parametric Toolbox 3.0

Martin Herceg
Manfred Morari

Michal Kvasnica

Colin N. Jones

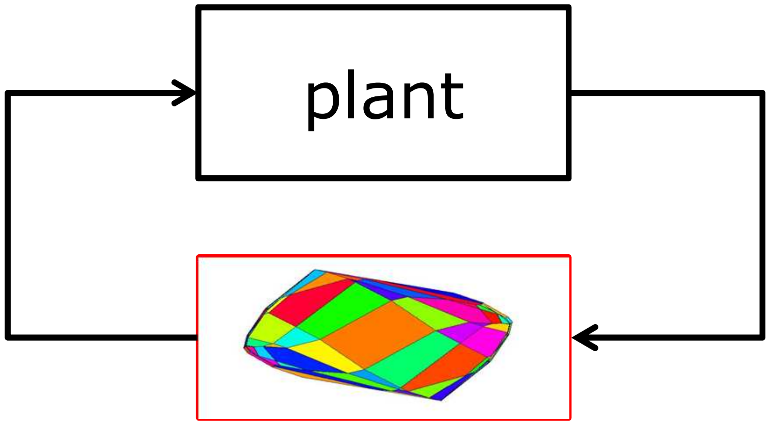
ETH Zürich

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

$$\begin{gathered}\min f^T x \\ Ax \leq b\end{gathered} \quad \Rightarrow$$

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

Parametric Optimization

- **Single solver** for parametric linear and quadratic problems

optimality conditions

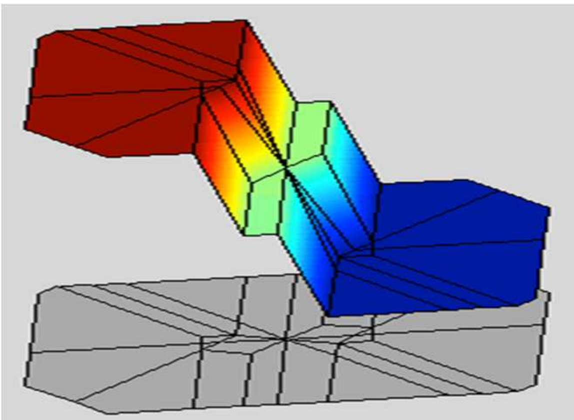
$$\min {}^1\!/_2 u^T H u + c^T u$$
$$Au \le b + E\xi,\ \ u \ge 0$$

$$Hu + c + A^T\lambda - \nu = 0$$
$$Au \le b + E\xi,\ \ u \ge 0$$
$$\lambda^T(Au - b - E\xi) = 0,\ \ \nu^T u = 0$$

LCP formulation

find $w,\ z$

$$w - Mz = q + G\xi$$
$$w \ge 0,\ \ z \ge 0$$
$$w^T z = 0$$



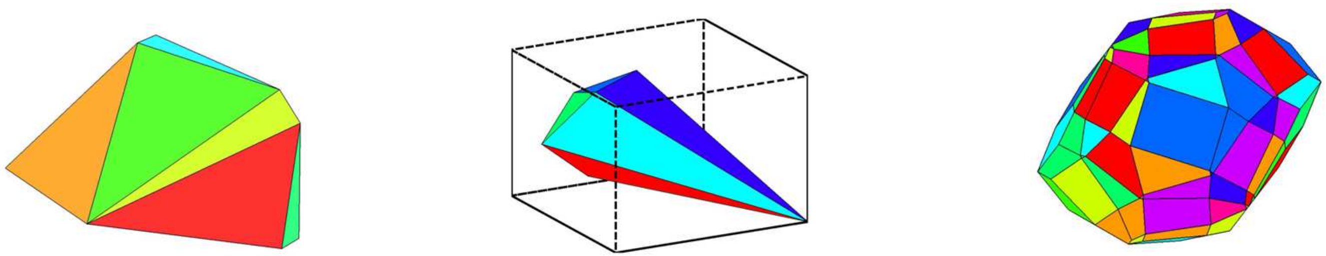
Geometric Library – Basic Sets

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

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

  - **unions of polyhedra** with certain properties

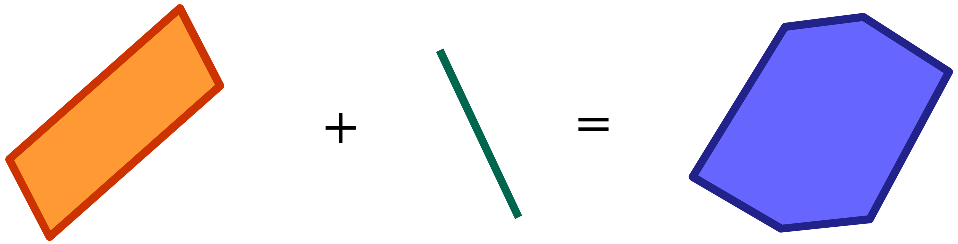
```
U = PolyUnion('Set', P, 'Convex', 1, 'Overlaps', 0)
```



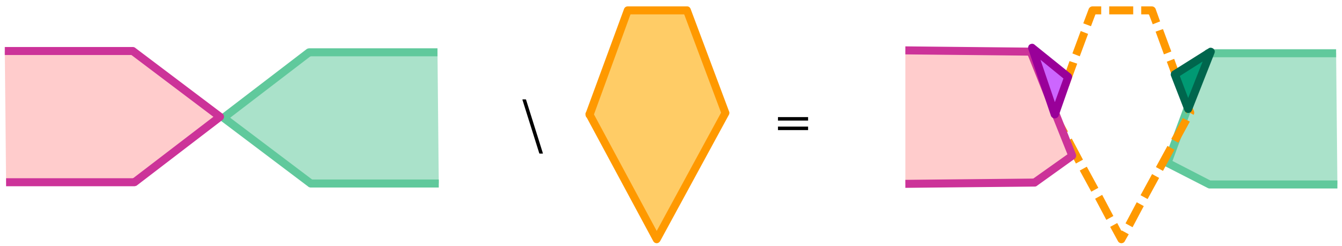
Geometric Library – Operations

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

```
S = P + Q
```



```
T = U \ S
```

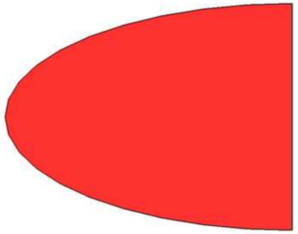


All algorithms built with MPT2 still work!

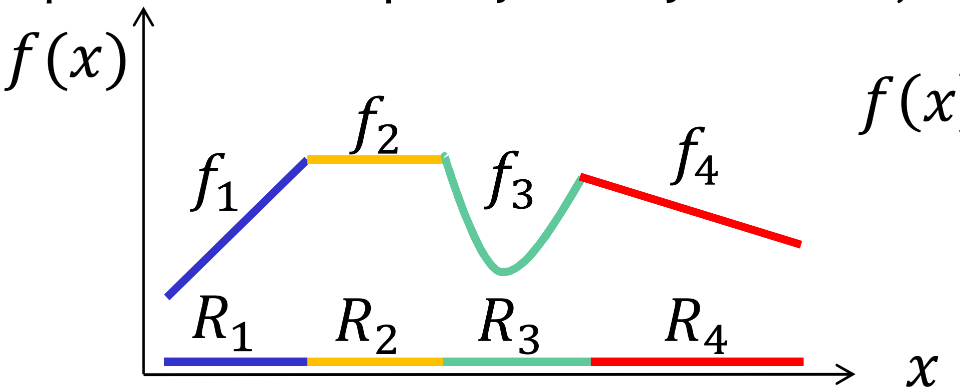
Geometric Library – Extensions

- **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



$$f(x) = f_i(x) \quad \text{if} \quad x \in R_i$$

New MPC Setup

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

$$\min \sum_{i=0}^{N} (\|Qx_k\| + \|Ru_k\|)$$
$$\text{s.t.: } x_{k+1} = f(x_k, u_k)$$
$$x \in \mathcal{X}, u \in \mathcal{U}$$

- 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 $x_{k+1} = Ax_k$

```
model = LTISystem('A', A)
```

- Affine autonomous system $x_{k+1} = Ax_k + f$

```
model = LTISystem('A', A, 'f', f)
```

- State update equation $x_{k+1} = Ax_k + Bu_k$

```
model = LTISystem('A', A, 'B', B)
```

- Output equation $x_{k+1} = Ax_k + Bu_k,\ y_k = Cx_k$

```
model = LTISystem('A', A, 'B', B, 'C', C)
```

Constraints

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

$$-1 \le u \le 1$$

```
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, …**)

$$x_N \in \Omega, \qquad \Omega = \{x \mid Ax \le b\}$$

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

Performance Specifications

- Penalties on system signals

$x^T Q x$

```
model.x.penalty = QuadFunction(Q)
```

$\|Qx\|_1$

```
model.x.penalty = OneNormFunction(Q)
```

$\|Qx\|_\infty$

```
model.x.penalty = InfNormFunction(Q)
```

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

MPT2 setups can be seamlessly imported!

```
model = mpt_import(sysStruct,probStruct)
```

Generation of Explicit Solution

1. **Construct the online MPC controller** object

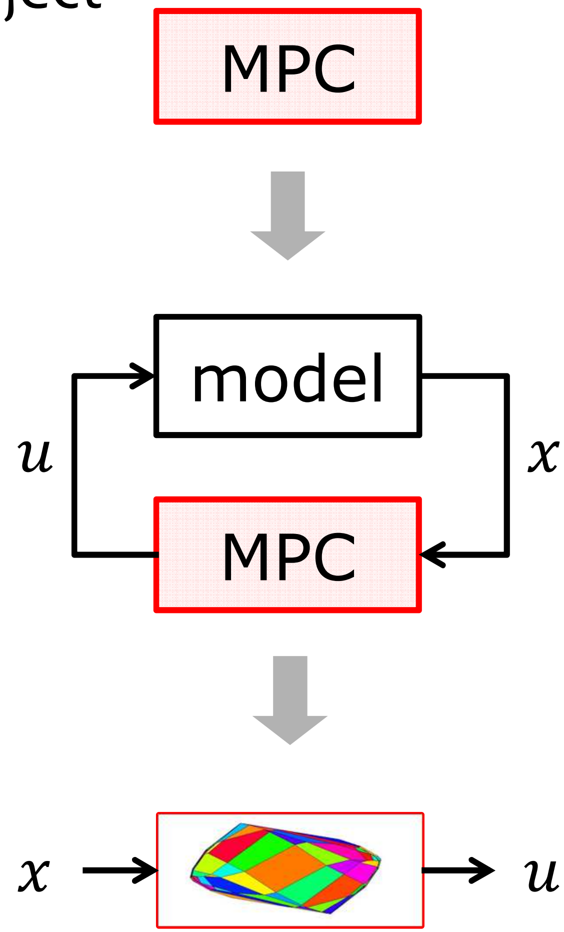
```
ctrl = MPCController(model, N)
u = ctrl.evaluate(x)
```

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()
```



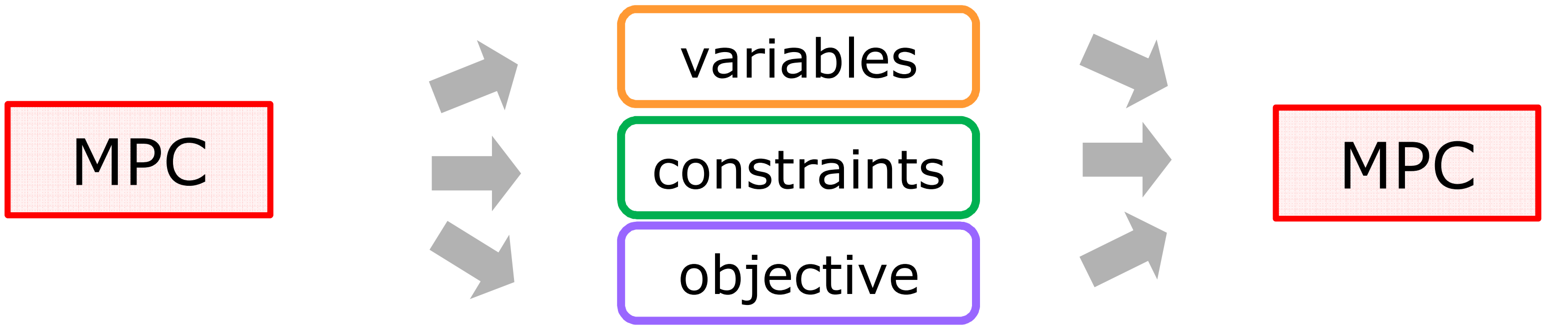
Fine Tuning

- **Tuning and refinement of MPC setups using YALMIP**
  - export to YALMIP

```
Y = ctrl.toYALMIP()
```

  - adjust constraints and performance specification
  - construct back the online MPC object

```
ctrl.fromYALMIP(Y)
```



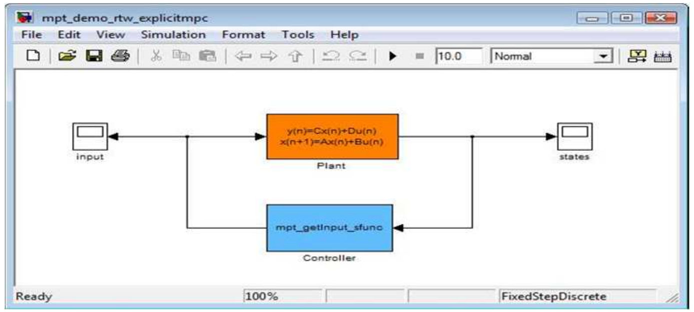
Arbitrary adjustmets are possible!

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



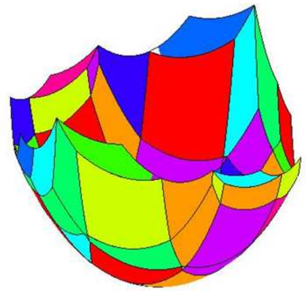
- Test in Simulink and deploy on real-time platform

C code

2D Example

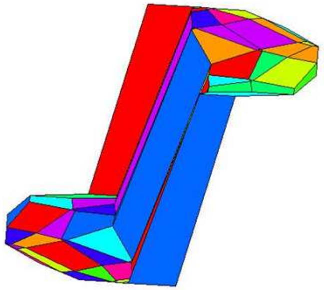
- **Oscillator example**
  - CFTOC with horizon 5
  - **cost function**

```
expl_ctrl.cost.fplot()
```



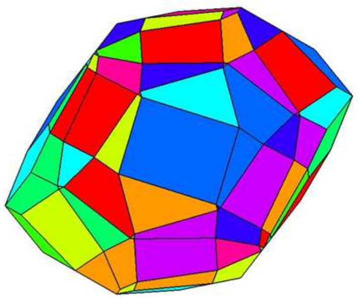
  - **feedback law**

```
expl_ctrl.feedback.fplot()
```



  - **partition**

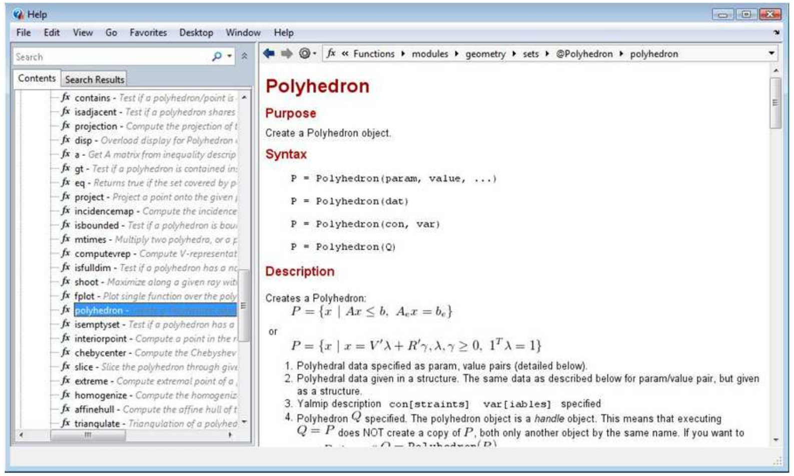
```
expl_ctrl.partition.plot()
```



Summary

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

```
mptdoc
```



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

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