# Efficient Nonlinear Model Predictive Control Algorithms

Mark Cannon
Department of Engineering Science, University of Oxford
mark.cannon@eng.ox.ac.uk

Nonlinear Predictive Control Workshop
CDC03



## NMPC development

NMPC can guarantee/quantify:

- Stability, feasibility regions
- Robustness
- Optimality

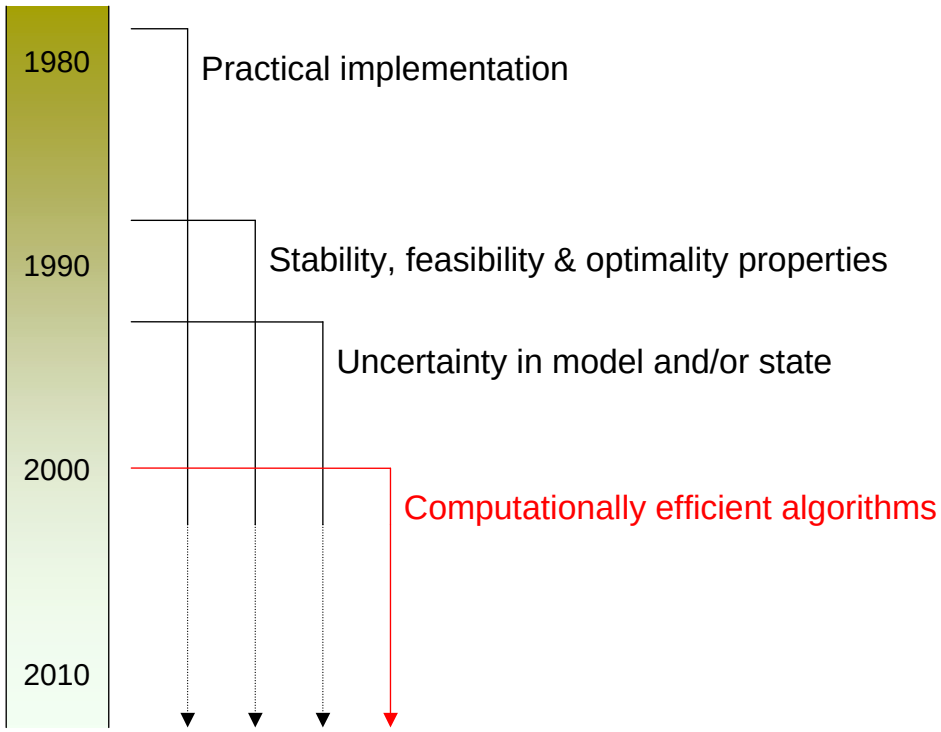
but

High computational complexity implies:

- Slow take-up by industry
- Impractical for fast dynamics

  e.g. aerospace, robotics, electromechanical applications

## NMPC development






## Efficient NMPC survey

1. Direct optimization methods
   - SQP : sequential & simultaneous
   - successive linearization
2. Optimal control approaches
   - Euler-Lagrange formulation
   - Approximate HJB solution
3. Cost and constraint modifications
   - Feasible sets & cost bounds determined offline
   - State-space partitioning
4. Re-parameterization of degrees of freedom in predictions
   - Reduced dimension optimization

## Problem formulation

- Plant model $x(k+1) = f(x(k), u(k))$

  $u(k) \in \mathcal{U}, \quad x(k) \in \mathcal{X}$

  constraint sets $\mathcal{U}, \mathcal{X}$ polytopic

- Quadratic cost

$$J(\mathbf{x}(k), \mathbf{u}(k)) = \sum_{i=0}^{N-1} \| x(k+i \mid k) \|_Q^2 + \| u(k+i \mid k) \|_R^2 \quad + \quad \| x(k+N \mid k) \|_{\bar{Q}}^2$$

$Q, R > 0$

- Receding horizon optimization

$$\boxed{\begin{aligned} \min_{\mathbf{u}(k), \mathbf{x}(k)} \quad & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \text{s.t.} \quad & x(k+i \mid k) \in \mathcal{X}, \quad u(k+i \mid k) \in \mathcal{U} \\ & x(k+i+1 \mid k) = f(x(k+i \mid k), u(k+i \mid k)), \quad i = 0, \ldots, N-1 \\ & x(k+N \mid k) \in \Omega_T \end{aligned}}$$

terminal weight $\bar{Q}$ and constraint set $\Omega_T$



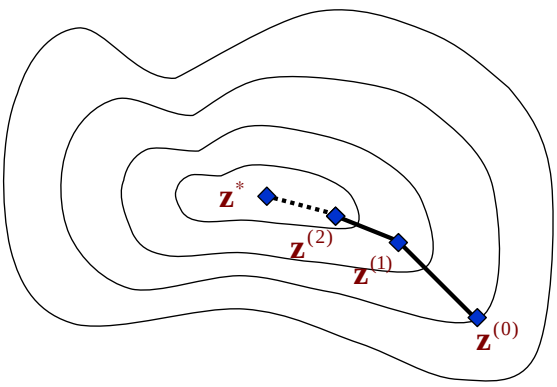
## 1. Direct optimization

Methods based on Successive Quadratic Programming (SQP)

- successive local approximations: quadratic cost, linear constraints
- QP problem solved at each iteration

e.g. $\min_{\mathbf{z}} f(\mathbf{z})$ (unconstrained):

contours of $f(\mathbf{z})$



## SQP

MPC optimization: $\min_{\mathbf{z}} \quad J(\mathbf{z})$

$$\text{s.t.} \quad c_E(\mathbf{z}) = 0, \quad c_I(\mathbf{z}) \le 0$$

predicted state/input trajectory: $\mathbf{z}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix}$

- Lagrangian: $\mathcal{L}(\mathbf{z}, \lambda, s) = J(\mathbf{z}) + \lambda^T \begin{bmatrix} c_E(\mathbf{z}) \\ c_I(\mathbf{z}) + s \end{bmatrix}$

  Lagrange multipliers: $\lambda = \begin{bmatrix} \lambda_E \\ \lambda_I \end{bmatrix}$

  slack variables: $s$

- KT conditions at solution:

$$\nabla \mathcal{L}(\mathbf{z}^*, \lambda^*, s^*) = 0, \quad s^* \ge 0, \quad \lambda_I^* \ge 0, \quad \lambda_I^{*T} s^* = 0 \qquad \nabla = \begin{bmatrix} \nabla_{\mathbf{z}} \\ \nabla_{\lambda} \end{bmatrix}$$

$$\downarrow$$

SQP solves $\nabla \mathcal{L} = 0$ for $\mathbf{z}^*, \lambda^*, s^*$ s.t. non-negativity & complementarity



## SQP iteration

Newton's method applied to $\nabla \mathcal{L}(\mathbf{z}, \lambda, s) = 0$

- *k*th iteration:

  (i). solve for $d^{(k)}, \delta^{(k)}$ satisfying

$$\nabla^2 \mathcal{L}(\mathbf{z}^{(k)}, \lambda^{(k)}, s^{(k+1)}) \begin{bmatrix} d^{(k)} \\ \delta^{(k)} \end{bmatrix} = -\nabla \mathcal{L}(\mathbf{z}^{(k)}, \lambda^{(k)}, s^{(k+1)})$$

$$s^{(k+1)} \ge 0, \quad \lambda^{(k)} + \delta^{(k)} \ge 0, \quad (\lambda^{(k)} + \delta^{(k)})^T s^{(k+1)} = 0$$

  (ii). set $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + d^{(k)}, \quad \lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)}$

- Uses quadratic approximation of $\mathcal{L}(\mathbf{z}, \lambda, s)$

$$\downarrow$$

reduce step size $d^{(k)}, \delta^{(k)}$ to limit approximation error

## SQP iteration

- Equivalent QP subproblem:

$$d^{(k)} = \arg\min_{d} \quad \nabla_z J(\mathbf{z}^{(k)})^T d + \tfrac{1}{2} d^T B(z^{(k)}, \lambda^{(k)}) d$$
$$\text{s.t.} \quad \nabla_z c_E(\mathbf{z}^{(k)})^T d + c_E(\mathbf{z}^{(k)}) = 0$$
$$\nabla_z c_I(\mathbf{z}^{(k)})^T d + c_I(\mathbf{z}^{(k)}) \le 0$$

- $d^{(k)} =$ search direction at *k*th iteration

  $\mathbf{z}^{(k+1)}$, $\lambda^{(k+1)}$ determined e.g. via line search

- $B(\mathbf{z}, \lambda) = \nabla_{\mathbf{z}}^2 \mathcal{L}(\mathbf{z}, \lambda)$
  $= \nabla_{\mathbf{z}}^2 J(\mathbf{z}) + \sum_i \lambda_i \nabla_z^2 c_i(\mathbf{z}, \lambda)$ $\Rightarrow$ quadratic local convergence rate $(\mathbf{z}^{(k)}, \lambda^{(k)}) \to (\mathbf{z}^*, \lambda^*)$

  or

  $B(\mathbf{z}, \lambda) =$ p.d. approximation to $\nabla_z^2 \mathcal{L}(\mathbf{z}, \lambda)$ $\Rightarrow$ slower convergence but convex QP



## Nonconvexity

Nonconvex NMPC optimization

$\Downarrow$

no guarantee of:

(i). convergence to global optimum

$\downarrow$

monotonicity of predicted cost – closed-loop stability?

(ii). number of iterations for given solution accuracy

$\downarrow$

computational complexity – implementable?

## SQP customization

- Warm starting:
  initialize solver at solution computed at previous sampling instant
  - Reduces computational complexity
  - Allows for monotonic cost guarantee

- Early termination:
  terminate solver at suboptimal feasible point
  - Allows for stability with only a single iteration at each sample

- Exploitation of subproblem structure:
  use efficient routines to factorize $\nabla^2 \mathcal{L}$
  - Reduces computational complexity of each iteration



## SQP solution methods

- Sequential approach:
  eliminate $c_E(\mathbf{z})$: simulate model dynamics at each iteration
  $\downarrow$
  feasible trajectory $\mathbf{x}$ available at each iteration
  $\Rightarrow$ warm starting & early termination possible

- Simultaneous methods:
  retain $c_E(\mathbf{z})$ : solve model dynamics for $\mathbf{x}$ & optimize $\mathbf{u}$ concurrently
  $\downarrow$
  $\nabla^2 \mathcal{L}$ almost block diagonal
  $\Rightarrow$ sparse factorization routines possible

  numerical discretization of continuous time dynamics
  incorporated efficiently

## Successive linearization

For $B(\mathbf{z}) = \nabla_{\mathbf{z}}^2 J(\mathbf{z})$ sequential SQP is equivalent to successive linearization:

- *k*th iteration

  (i). linearize model dynamics about trajectory $(\mathbf{x}^{(k)}, \mathbf{u}^{(k)})$

  (ii). update $\mathbf{u}^{(k+1)}$ using search direction:

$$d_{\mathbf{u}}^{(k)} = \arg\min_{d_{\mathbf{u}}} J_{LIN}(d_{\mathbf{u}}) \text{ s.t. } c_{LIN}(d_{\mathbf{u}}) \le 0$$

- Linear uncertain MPC optimization at each iteration

- Need to account for linearization errors to ensure
  - sufficient cost reduction: $J(\mathbf{z}(k)) - J(\mathbf{z}(k+1))$
  - constraint satisfaction: $c_I(\mathbf{z}(k)) \le 0$, $c_E(z(k)) = 0$



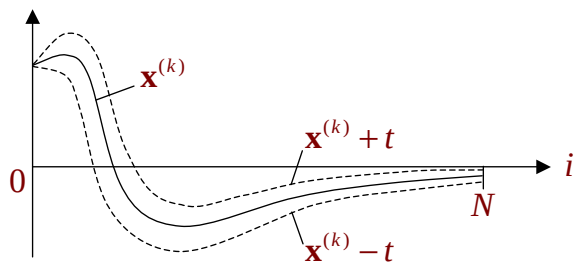
## Handling linearization errors

- Variable trajectory bounds: $t \ge |x^{(k+1)} - x^{(k)}|$ [Lee et al. 2003]

  - cost & constraint upper bounds

$$J_{LIN}(d_{\mathbf{u}}, t) \ge J(d_{\mathbf{u}}), \quad c_{LIN}(d_{\mathbf{u}}, t) \ge c_I(d_{\mathbf{u}})$$

$$t = 0 \Rightarrow J_{LIN} = J, \; c_{LIN} = c$$




  - SQP iteration: $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + d_{\mathbf{u}}^{(k)}$

$$(d_{\mathbf{u}}^{(k)}, t^*) = \arg\min_{d_{\mathbf{u}}, t} J_{LIN}(d_{\mathbf{u}}, t)$$

$$\text{s.t.} \quad c_{LIN}(d_{\mathbf{u}}, t) \le 0$$

$$|d_{\mathbf{x}}| \le t$$

- Single SQP iteration $\longrightarrow$ closed-loop stability & convergence

## SQP Summary

| | sequential | simultaneous |
|---|---|---|
| warm starting & early termination | ✓ | × |
| continuous-time models | × | ✓ |
| Large-scale problems | × | ✓ |



## 2. Optimal control approaches

- Aim: avoid optimality vs. computational efficiency trade-off
  - optimal control law incorporated in predictions
  - increased offline pre-processing
- Euler-Lagrange formulation
  - solve approximately for optimal control law online
- Hamilton-Jacobi-Bellman formulation
  - solve HJB equation for optimal cost offline

## Problem formulation

- Continuous-time model dynamics

$$\dot{x}(t) = f(x(t),u(t))$$
$$u(t) \in \mathcal{U}$$

- Predicted performance index at time $t$ :

$$J(t) = \int_0^T l(x(\tau,t),u(\tau,t))\, d\tau \; + \; \phi(x(T,t))$$

$x(\tau,t), u(\tau,t) =$ predicted $x(t+\tau),\; u(t+\tau)$ at time $t$

- MPC optimization:

$$\boxed{\begin{aligned} \min_{u(\tau,t),\ \tau\in[0,T]} \quad & J(t) \\ \text{s.t.} \quad & \frac{\partial}{\partial\tau} x(\tau,t) = f(x(\tau,t),u(\tau,t)) \\ & c(u(\tau,t)) \le 0 \end{aligned}}$$

$c(u) \le 0 \;\Leftrightarrow\; u \in \mathcal{U}$



## Euler-Lagrange approach

- Hamiltonian: $\mathcal{H}(x,u,\lambda,\mu) = l(x,u) + \lambda^T f(x,u) + \mu^T c(u)$

$\lambda(\tau,t)$ : costate

$\mu(\tau,t)$ : multiplier for $c(u) \le 0$

- Conditions for optimality (E-L eqn.s)

$$\boxed{\begin{aligned} & \frac{\partial}{\partial\tau} x(\tau,t) = \nabla_\lambda \mathcal{H} && x(0,t) = x(t) \\ & \frac{\partial}{\partial\tau} \lambda(\tau,t) = -\nabla_x \mathcal{H} && \lambda(T,t) = \nabla_x \phi(x(T,t)) \\ & \nabla_u \mathcal{H} = 0 && \mu(\tau,t) \begin{cases} = 0 & \text{if } c(u(\tau,t)) \le 0 \\ \ge 0 & \text{if } c(u(\tau,t)) = 0 \end{cases} \end{aligned}}$$

$\downarrow$

- $u^*(\tau,t)$ obtained by simulation if $\lambda^*(0,t)$ known

- $\lambda^*(0,t)$ unknown $\Rightarrow$ two-point boundary value problem (difficult)

## Continuation

- Terminal condition on $\lambda$ :

$$\lambda(T,t) - \nabla_x \phi(x(T,t)) = F(\lambda(0,t), x(t), T) = 0$$

- Compute $\lambda(0,t)$ satisfying $F = 0$ by continuation [Ohtsuka & Fujii, 1997]

Summary of method

$T = T(t)$ : time-varying horizon

$$\left.\begin{array}{ll} \text{at } t = t_0 & : \; F(\lambda(0,t_0), x(t_0), T(t_0)) = 0 \\ \forall t \in [t_0, t_1] & : \; \dfrac{d}{dt} F(\lambda(0,t), x(t), T(t)) = 0 \end{array}\right\} \Rightarrow F(\lambda(0,t_1), x(t_1), T(t_1)) = 0$$

Initialize with $T(0) = 0$ :

$$F(\lambda(0,0), x(0), 0) = 0 \;\Rightarrow\; \lambda(0,0) = \nabla_x \phi(x(0))$$

(trivial solution)

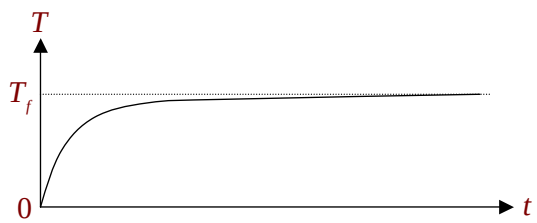


## Euler-Lagrange approach

- Offline – Construct smooth $T(t)$:

$T(0) = 0$

$T(t) \to T_f$ as $t \to \infty$




- Online –

(i). E-L eqn.s $\longrightarrow$ evaluate $x(T,t),\; \lambda(T,t),\; \dfrac{\partial x(T,t)}{\partial x(0,t)},\; \dfrac{\partial \lambda(T,t)}{\partial \lambda(0,t)} \ldots$

(ii). Update $\lambda(0,t)$:

$$\frac{dF}{dt} = 0 \;\Rightarrow\; \frac{d}{dt}\lambda(0,t) = -(\nabla_\lambda F)^{-1}\left[\nabla_x F \frac{d}{dt} x(t) + \frac{\partial F}{\partial T}\frac{d}{dt} T(t) + \zeta F\right]$$

(iii). E-L eqn.s $\longrightarrow$ $u(t) = u(0,t)$

## Euler-Lagrange approach

Disadvantages:

- No terminal state constraint
- Increasing $T(t)$ incompatible with state/input constraints

$\downarrow$

No stability/convergence guarantees



## HJB approach

- Cost-to-go at $t+\tau$ along $(x(\tau,t),u(\tau,t))$ :

$$\hat{J}(x(\tau,t),\tau)=\int_{\tau}^{T} l(x(s,t),u(s,t))\,ds \;+\; \phi(x(T,t))$$

- Optimal cost: $J^*(x)=\hat{J}^*(x,0)$ satisfies HJB eqn

$$\min_{u\in\mathcal{U}}\left\{\nabla_x \hat{J}^*(x,\tau)^T f(x,u)+l(x,u)\right\}=-\frac{\partial}{\partial\tau}\hat{J}^*(x,\tau)$$

$\uparrow$

Nonlinear PDE

$x(T,t)\in\Omega_T$ incorporated via B.C.s

- Offline – solve for $J^*(x)$ (difficult)

  Online – $u(t)=$ minimizing argument

## Iterative HJB solution method

[Saridis & Lee 1979]

- Initialize with $u = u^{(0)}(x)$ stabilizing for $x \in \Omega$
- For $k = 0,1,\ldots$ :

  (i). solve for $J^{(k+1)}(x)$ :

$$\nabla_x J^{(k+1)}(x) f(x, u^{(k)}(x)) + l(x, u^{(k)}(x)) = 0 \quad \forall x \in \Omega$$

  (ii). compute $u^{(k+1)}(x)$ :

$$u^{(k+1)}(x) = \arg\min_{u \in \mathcal{U}} \left\{ \nabla_x J^{(k+1)}(x) f(x,u) + l(x,u) \right\} \quad \forall x \in \Omega$$

- Linear PDE solved at each iteration

| | | |
|---|---|---|
| $J^{(k+1)}(x) =$ I.H. cost under $u = u^{(k)}(x)$ | $\Rightarrow$ | $u^{(k)}(x)$ stabilizing for $x \in \Omega$ |
| $J^{(k+1)}(x) \le J^{(k)}(x) \quad \forall x \in \Omega$ | | $(J^{(k)}, u^{(k)}) \to (J^*, u^*)$ |



## HJB approach

Disadvantages:

- PDE solved at each iteration of offline procedure
- High storage requirements for RH control law

$\downarrow$

Computation & storage prohibitive

## 3. Cost and constraint approximation

- Sampling interval: $T_{\text{samp}} \sim O(10^{-3})$ sec

  ↓

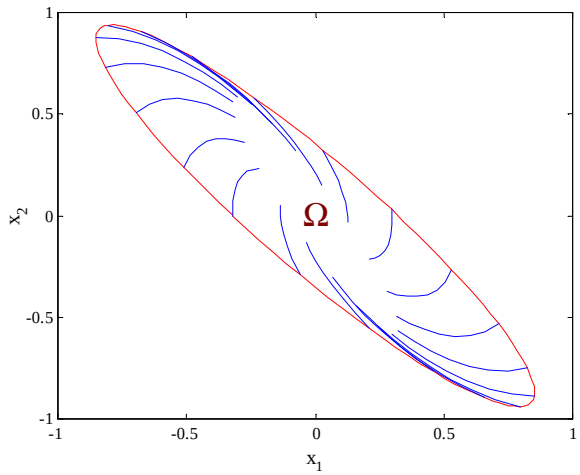
  Online computation $\leq$ small LP or convex QP

- Suboptimal MPC formulation
  - Approximate constraints using invariant or reachable sets
  - Approximate dynamics using ADI



## Invariant sets

- $\Omega$ is invariant under $\begin{cases} x(k+1) = f(x(k), u(k)) \\ u(k) = K(x(k)) \end{cases}$ iff $f(x, K(x)) \in \Omega \quad \forall x \in \Omega$

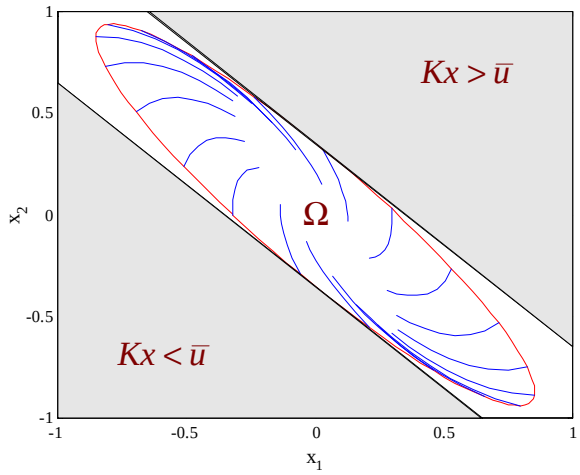



- $x(k) \in \Omega \quad \Rightarrow \quad x(k+i) \in \Omega \quad i = 1, 2, \ldots$

## Invariant sets

- $\Omega$ is feasible w.r.t. $x \in \mathcal{X}$ if $\Omega \subseteq \mathcal{X}$

  e.g. $\mathcal{X} = \{ x : |Kx| \le \bar{u} \}$ :



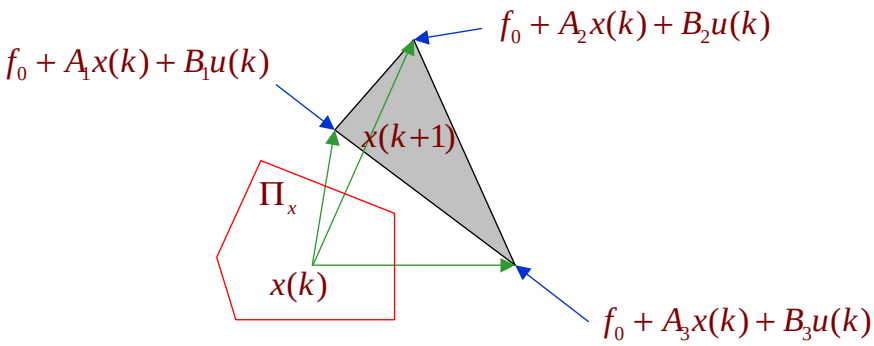

- $x(k) \in \Omega \quad \Rightarrow \quad |Kx(k+i)| \le \bar{u} \quad i = 1, 2, \ldots$



## Affine difference inclusion

- Local model representation as uncertain time-varying affine system:

$$f(x,u) \in Co\{ f_0 + A_i x + B_i u, \; i = 1, \ldots p \} \quad \forall x \in \Pi_x, \; u \in \Pi_u$$



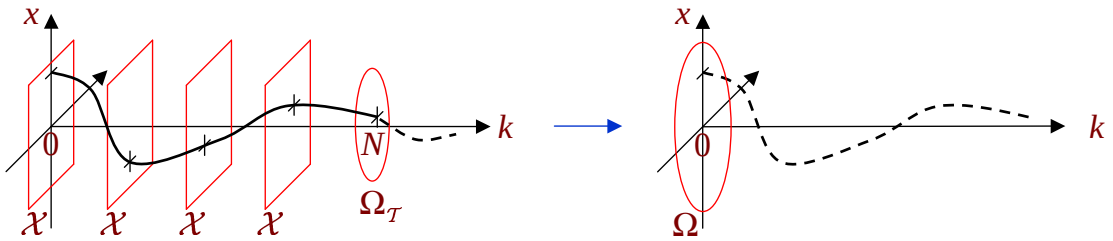

- Can construct ADI if $f$ continuously differentiable

$\downarrow$

- $x(k+1) \in$ ellipsoid or polytope $\Leftarrow$ linear or quadratic constraints on $x(k), u(k)$

## Implicit prediction horizon

- Nonlinear prediction dynamics $\Rightarrow$ nonconvex cost and constraints




- Constraint approximation under $u(k+i\,|\,k) = Kx(k+i\,|\,k)$ :

$$x(k) \in \Omega \quad \longleftarrow \quad \text{invariant, feasible}$$

- Use ADI $f(x, Kx) \in Co\{\,(A_i + B_i K)x\,\}$ to compute online:

$$(K(k), \Omega(k)) = \arg\min_{K,\Omega} \; \bar{J}(x(k)) \quad \text{s.t. invariance, feasibility}$$

[e.g. Kothare 96]

$\uparrow$

convex SDP



## Implicit prediction horizon

- Formulate d.o.f. as state of dynamic feedback law [Cannon, Kouvaritakis, 01]

$\downarrow$

SDP transferred to offline computation

- Optimization variables: $\{c(i\,|\,k),\; i = 0,\ldots N-1\}$

$$u(k+i\,|\,k) = \begin{cases} \kappa(x(k+i\,|\,k)) + c(i\,|\,k), & i = 0,\ldots,N-1 \\ \kappa(x(k+i\,|\,k)), & i \geq N \end{cases}$$

$\kappa$ fixed, stabilizing/optimal for no constraints

- Autonomous prediction dynamics:

$$z(k+1\,|\,k) = \begin{bmatrix} f(\,x(k), \kappa(x(k)) + c(0\,|\,k)\,) \\ M\mathbf{c}(k) \end{bmatrix}, \quad z(k) = \begin{bmatrix} x(k) \\ \mathbf{c}(k) \end{bmatrix}$$

## Implicit prediction horizon

- Offline – use ADI to compute $\Omega = \{ z : z^T P z \le 1 \}$ via

  $\max_P vol(\Omega_x)$ s.t. invariance, feasibility ← convex SDP

- Online – $\mathbf{c}(k) = \arg\min_{\mathbf{c}} \| c \|_W$ s.t. $\begin{bmatrix} x(k) \\ \mathbf{c} \end{bmatrix} \in \Omega$

  ↓

- Univariate online optimization, $O(N)$ N-R iteration
- Asymptotically stabilizing
- Finite-time convergence $u(k) \to \kappa(x(k))$



## Polytopic invariant sets

- Ellipsoidal constraint approximation ⇒ conservative region of attraction
- Reduce conservativeness through:
  - Polytopic invariant set: $\Omega = \{ x : Vx \le \underline{1} \}$

    vertices: $\{ x_i,\ i = 1, \ldots, m \}$ , $m$ arbitrary, fixed
  - Nonlinear feedback law in invariance condition

    e.g. $\kappa(x) = \sum_{i=1}^{m} \alpha_i u_i$ , $\{ u_i,\ i = 1, \ldots m \} =$ "controls at vertices"
  - Local ADI, $\Pi_x, \Pi_u$ computed simultaneously with $\Omega$

## Polytopic invariant sets

- Sequence of reachable sets $\Omega^{(1)},\ldots,\Omega^{(N)}$: [Cannon 03]

  $\Omega^{(N)}$ invariant

  $$\left.\begin{array}{r} f(x,\kappa(x))\in\Omega^{(k+1)} \\ \kappa(x)\in\mathcal{U}\end{array}\right\} \quad \forall x\in\Omega^{(k)} \longleftarrow \text{linear constraints on } \{(x_i^{(k)},u_i^{(k)})\}$$

  $$\Downarrow$$

  $\max vol(\Omega^{(k)})$ determined by sequence of LP's

- Offline – compute $\Omega^{(1)},\ldots,\Omega^{(N)}$

- Online – minimize cost bound:

  $$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \sum \alpha_i \quad \text{s.t. } x(k)=\sum \alpha_i x_i^{(k)},\ \alpha_i \geq 0 \longleftarrow \text{LP}$$

  or

  – incorporate $\kappa(x)$, $\bigcup_{k=1}^{N}\Omega^{(k)}$ in NMPC as terminal feedback & region



## State space partitioning

[Bacic 03]

- Improve cost approximation through offline dynamic programming

- Partition operating region into simplexes $\{S_i\}$ optimized sequentially:

  $$\max_{S_i} vol(S_i) - \lambda \max_{x\in S_i,\, u\in\mathcal{U}} \{\, \|f(x,u)\|_Q^2 + \|u\|_R^2 \quad \text{s.t. } f(x,u)\in\Omega^{(i)} \,\}$$

  local ADI $\Rightarrow$ QP problem

- Form weighted directed graph $G=\langle\{S_i\},\{T_{ij}\}\rangle$

  weights $\{w_{ij}\}$, $w_{ij}=$ stage cost of transition $T_{ij}$

  $$\downarrow$$

  optimal path $T_{ij}^*$ from $S_i$ determined by Floyd's algorithm

## State space partitioning

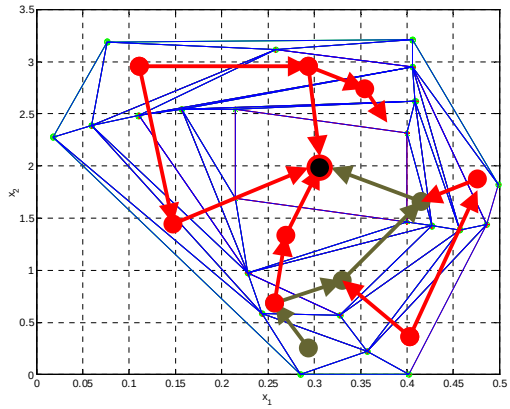
- Offline –

  (i). Partition operating region

  (ii). Construct weighted graph

  (iii). Compute optimal path from each simplex

CSTR example






## State space partitioning

- Online – if $x(k) \in S_i$ and optimal transition is $T_{ij}^*$ , then solve

$$u(k) = \arg\min_{u \in \mathcal{U}} \|f(x,u)\|_Q^2 + \|u\|_R^2 \quad \text{s.t.} \quad f(x,u) \in S_j$$

↓

- Single stage cost in online optimization

- Approximate solution of HJB equation

- Trade-off: performance vs. complexity of partition

  stability guarantee retained

## 4. Prediction parameterization

- Conventional MPC formulation: d.o.f. $= \{ u(k|k), \ldots, u(k+N-1|k) \}$

$$\left\{ \begin{array}{c} \text{large stabilizable set} \\ \text{good performance} \end{array} \right\}$$

$$\Uparrow$$

long horizon

$$\Downarrow$$

high online computational load

- Re-parameterize predictions to avoid trade-off



## Interpolation MPC

- Univariate predictions: [Kouvaritakis, 98]

$$u(k+i|k) = (1-c(k))\kappa^o(x(k+i|k)) + c(k)\kappa^f(x(k+i|k))$$

- $c(k)$ : d.o.f. at time $k$
- $\kappa^o$ : optimal for no constraints
- $\kappa^f$ : large stabilizable set, poor performance

$$\downarrow$$

- Prediction class inherits: optimality of $\kappa^o$ + stabilizable set of $\kappa^f$

## Interpolation MPC

- Online optimization:

$$c(k) = \arg\min_{c} \; J(k) \quad \text{s.t.} \quad u(k\,|\,k) \in \mathcal{U}$$
$$x(k+1\,|\,k) \in \Omega_T$$
$$\Psi(x(k+1\,|\,k)) - \Psi(x(k)) \le -l(x(k),u(k))$$

- Objective: $J(k) = \sum_{i=0}^{N-1} l(x(k+i\,|\,k),u(k+i\,|\,k)) + \Psi(x(k+N\,|\,k))$

- Or use objective: $\min_{c} c$ ← reduced computation

- Stability constraints:
  - recursion of feasibility
  - closed-loop convergence $l(x(k),u(k)) \to 0$



## 2-dimensional formulation

- Interpolate over $\kappa^o,\ \kappa^f,\ u^t$ [Kouvaritakis, 00]

- Tail trajectory:

$$u^t(k+i\,|\,k) = \begin{cases} u(k+i\,|\,k-1) & i = 0,\ldots,N-2 \\ \kappa^f(x(k+i\,|\,k)) & i = N-1 \end{cases}$$

implement as feedback law using $c(k-1)$

- Conventional cost and constraints:

$$c(k) = \arg\min_{c} \; J(k) \quad \text{s.t.} \quad u(k+i\,|\,k) \in \mathcal{U},\ x(k+i\,|\,k) \in \mathcal{X},\ i = 0,\ldots,N-1$$
$$x(k+N\,|\,k) \in \Omega_T$$

↑

explicit prediction horizon $N \ge 2$

## Higher dimensional formulation

- Input predictions: [Magni, 01]

$$\{\underbrace{u(k\,|\,k),\ldots,u(k+N_c-1\,|\,k)}_{\text{d.o.f.}},\kappa^f(x(k+N_c\,|\,k)),\ldots,\kappa^f(x(k+N-1\,|\,k))\}$$

- Conventional cost and constraints:

$$\mathbf{u}(k)=\arg\min_{\mathbf{u}}\ J(k)\quad \text{s.t.}\ u(k+i\,|\,k)\in\mathcal{U},\ x(k+i\,|\,k)\in\mathcal{X},\ i=0,\ldots,N-1$$
$$x(k+N\,|\,k)\in\Omega_T$$

- Stabilizable set increases with $N$
- Large $N$ is computationally cheap if $N_c$ is small



## Summary

1. Direct optimization methods – SQP
2. Optimal control approaches – E-L, HJB
3. Cost and constraint approximations – invariant sets, ADI
4. Re-parameterization of degrees of freedom in predictions – interpolation

# Summary

- Direct optimization: $T_{samp} > 1$ sec

- Univariate algorithms: $T_{samp} > 10^{-3}$ sec

$\downarrow$

- For fast dynamics need

  explicit prediction horizon $N \geq 1$ and $T_{samp} < 10^{-3}$ sec