Optimization-based control of constrained nonlinear systems:

When should we use continuous-time models?

Fernando A. C. C. Fontes

Universidade do Porto

faf@fe.up.pt

http://www.fe.up.pt/~faf

(ffontes@tamu.edu, Wisenbaker 333B)

Texas A&M, March 25, 2016
University of Porto

- 30,000 students (8,000 engineering, 1,100 ECE)
- 20%~25% of the Portuguese scientific production
- In most rankings: 1st Portuguese University, Top 100 European, Top 10 Ibero-American
About Porto

- On the Atlantic coast
- 1.5 M people (urban area)
- Best known for FC Porto, Port Wine, Douro wine region
About Porto

- On the Atlantic coast
- 1.5 M people (urban area)
- Best known for FC Porto, Port Wine, Douro wine region
Optimization-based control of constrained systems: When should we use continuous-time models?

About Porto

- On the Atlantic coast
- 1.5 M people (urban area)
- Best known for FC Porto, Port Wine, Douro wine region
About Porto

- On the Atlantic coast
- 1.5 M people (urban area)
- Best known for FC Porto, Port Wine, Douro wine region
Outline I

Part I: The Sampled-data MPC framework

Part II: Phenomena best studied in continuous-time

- Stability
- Discontinuous Feedbacks
- Bang-bang control
- Path-following
- Impulsive systems

Part III: Efficient solution of Continuous-Time optimal control problems

- Numerical Solution of OCP
- Mesh Refinement for Optimal Control
- Mesh refinement in MPC
Part I

The Sampled-data MPC framework
Control of Complex Systems involves

- Feedback
- Planning (not just reacting to the error),
- Discretization (a computer in the loop),
- Optimization (enhance performance, deal with constraints)

Optimization-based control with discrete-time models would (apparently) seem more natural.

When should we use continuous-time models?
Model Predictive Control: Main Idea

The Receding Horizon Strategy

To satisfy long term objectives, using short term planning repeatedly.

By re-planning repeatedly, taking into account the current state, we obtain a feedback strategy

Examples:
Chess game – we can only plan a few moves ahead, not the whole game
Car driving – we often only plan the trajectory within our sight

Importance of planning ahead + feedback
Model Predictive Control: Main Idea

The Receding Horizon Strategy

To satisfy long term objectives, using short term planning repeatedly.

By re-planning repeatedly, taking into account the current state, we obtain a feedback strategy

Examples:
Chess game – we can only plan a few moves ahead, not the whole game
Car driving – we often only plan the trajectory within our sight

Importance of planning ahead + feedback
Model Predictive Control: Main Idea

The Receding Horizon Strategy

*To satisfy long term objectives, using short term planning repeatedly.*

By re-planning repeatedly, taking into account the current state, we obtain a *feedback strategy*

**Examples:**
Chess game – we can only plan a few moves ahead, not the whole game
Car driving – we often only plan the trajectory within our sight
Model Predictive Control: Main Idea

The Receding Horizon Strategy

*To satisfy long term objectives, using short term planning repeatedly.*

By re-planning repeatedly, taking into account the current state, we obtain a *feedback strategy*.

**Examples:**
Chess game – we can only plan a few moves ahead, not the whole game
Car driving – we often only plan the trajectory within our sight

Importance of planning ahead + feedback
Model Predictive Control: Main Idea

Objective: Obtain a closed-loop control, a map $x \mapsto k(x)$, for all states $x$ in the trajectory ... optimizing certain criteria.

Fact 1: Solving a closed-loop OCP (optimal control problem) could give us the solution. BUT it’s generally very hard except in special cases like LQR.

Fact 2: Open-loop OCPs, obtaining a map $t \mapsto u(t)$, can be solved much more efficiently; several solvers available. There’s a body of theory to deal with virtually any problem (state-contraints, nonsmothness, ...)

The MPC algorithm:

1. Solve on-line a sequence of open-loop OCPs with a fixed short horizon. Each problem uses as initial state the current measured state of the true trajectory.
2. Concatenate the initial parts of each control to obtain a “feedback”.
The sampled-data MPC framework

Consider a sequence \( \{t_i\}_{i \geq 0} \) s.t. \( t_{i+1} = t_i + \delta, \delta > 0 \):

1. Measure state of the plant \( x_{t_i} \)
2. Get \( \bar{u} : [t_i, t_i + T] \mapsto \mathbb{R}^m \) solution to the OCP:

\[
\text{Minimize} \quad \int_{t_i}^{t_i+T} L(t, x(t), u(t)) dt + W(x(t_i + T))
\]

subject to

\[
\dot{x}(t) = f(t, x(t), u(t)) \quad \text{a.e.} \ t \in [t_i, t_i + T] \\
x(t_i) = x_{t_i} \\
u(t) \in U(t) \quad \text{a.e.} \ t \in [t_i, t_i + T] \\
x(t) \in X \quad \text{a.e.} \ t \in [t_i, t_i + T] \\
x(t_i + T) \in S
\]

3. Apply to the plant the control \( u^*(t) := \bar{u}(t) \) in the interval \([t_i, t_i + \delta]\). (the remaining control \( \bar{u}(t) \), \( t > t_i + \delta \) is discarded)

4. Repeat for the next sampling time \( t_i = t_i + \delta \)

\( \{ t_i \} \) Sequence of sampling instants
\( \delta \) Sampling interval
\( x_{t_i} \) Plant state measured at time \( t_i \).
\( (\bar{x}, \bar{u}) \) OL state/control solving the OCP
\( (x^*, u^*) \) CL state/control of the MPC
\( T, L, W, S \) Design parameters
Industrial impact

- Long history in the process industry (Petrochemical 1970’s)
- Several commercial products
- Thousands of applications (surveys to software vendors identified 2200 applications in 1996 and 5000 in 2000 [QB03])
- Advanced technique well-accepted in industry
- Strong economic benefits
- Application of MPC to fast-systems is growing in the last few years (robotics, cars, aeronautics, ... elect. eng. devices)
Phenomena best studied in continuous-time

Part II

Phenomena best studied in continuous-time
Phenomena best studied in continuous-time

Stability

Discontinuous Feedbacks

Bang-bang control

Path-following

Impulsive systems
Phenomena best studied in continuous-time

Stability

Discrete-time models and stability

Discrete-time models might not capture inter-sample behaviour

(e.g. designing overly aggressive dead-beat controllers might lead to hidden inter-sample oscillations or even instability)
**MPC and stability**

OL-OCP drives state to the origin \( \not\Rightarrow \) MPC trajectory goes to the origin

Minimize \( \int_0^{10} u^2(t) dt \)

Subject to

\[
\begin{align*}
\dot{x} &= v \cos(\psi) & \text{a.e. } t \in [0, 10] \\
\dot{y} &= v \sin(\psi) & \text{a.e. } t \in [0, 10] \\
\dot{\psi} &= w & \text{a.e. } t \in [0, 10]
\end{align*}
\]

\( 0 \leq v(t) \leq 1 \) \( \text{a.e. } t \in [0, 10] \)

\( -0.7 \leq w(t) \leq 0.7 \) \( \text{a.e. } t \in [0, 10] \)

\( x_0 = (-1, -2, -\pi/4) \)

\( x_f = (0, 0, 0) \)

---

Open-loop OCP, *** Closed-loop MPC

All computed open-loop OC trajectories go to the origin, BUT the MPC trajectory does not!
MPC schemes with guaranteed stability

[Discrete-time] and [Continuous-time]

- Infinite Horizon (p.d. running cost, finite cost ⇒ stability)
- Terminal Equality Constraint [Kerthy, Gilbert 88] [Mayne, Michalska 90]
  \[ S = \{0\}\]
- Contraction Constraints [Yang, Polak 93] [DeOliveira, Morari 98].
  \[ S_{x_0} := \{x_f : x_f^T P x_f \leq \alpha x_0^T P x_0\}, \quad \alpha \in (0, 1) .\]
- Dual-Mode [Michalska, Mayne 93]
  First \( S = \epsilon B, \ T \) free. Then use linear feedback.
- Terminal Cost based:
  - Linear case [Rawlings, Muske, 93].
    \[ L = x^T Q x + u^T R u, \quad W = x^T P x, \quad P \text{ solves an ARE} \]
  - Quasi-infinite Horizon [Chen, Allgower 98], [DeNicolao et al. 98]
    \[ S = \epsilon B, \quad W = \int_T^\infty L(x(t), \tilde{u}(t)) dt, \quad \tilde{u} \text{ is a linear feedback.} \]
- Global CLF approaches [Jadbabaie et al. 99, Primbs et al. 00]
  A global Lyapunov function has to be known
- **Sufficient Stability Condition on the Design Parameters (Local CLF):**
  [Mayne et al. 2000], [Fontes 99, Fontes 2001].
- Unconstrained MPC (no terminal cost, sufficiently long horizons)
  [Grune 2009], [Reble Allgower 2012]
Phenomena best studied in continuous-time
Stability

The MPC framework

Consider a sequence \( \{t_i\}_{i \geq 0} \) s.t. \( t_{i+1} = t_i + \delta, \delta > 0 \):

1. Measure state of the plant \( x_{t_i} \)
2. Get \( \bar{u} : [t_i, t_i + T] \mapsto \mathbb{R}^m \) solution to the OCP:

Minimize \( \int_{t_i}^{t_i+T} L(t, x(t), u(t)) dt + W(x(t_i + T)) \)
subject to

\[
\begin{align*}
\dot{x}(t) &= f(t, x(t), u(t)) & \text{a.e. } t \in [t_i, t_i + T] \\
x(t_i) &= x_{t_i} \\
u(t) &\in U(t) & \text{a.e. } t \in [t_i, t_i + T] \\
x(t) &\in X & \text{a.e. } t \in [t_i, t_i + T] \\
x(t_i + T) &\in S
\end{align*}
\]

3. Apply to the plant the control \( u^*(t) := \bar{u}(t) \) in the interval \([t_i, t_i + \delta]\). (the remaining control \( \bar{u}(t), t > t_i + \delta \) is discarded)

4. Repeat for the next sampling time \( t_i = t_i + \delta \)
A Sufficient Condition for Stability

Choose the design parameters: time horizon $T$, objective functions $L$ and $W$, and terminal constraint set $S$, to satisfy:

**SC1–SC4** ... $S$ is closed, contains the origin is reachable in time $T$ ... $W$ is locally Lipschitz continuous and positive definite ...

**SC5** $\forall x_t \in S$, $\exists \tilde{u} : [t, t + \delta] \to U$ [or $\tilde{u}_t \in U$] such that

$$W(x(t + \delta; t, x_t, \tilde{u})) - W(x_t) \leq -\int_t^{t+\delta} L(x(s), \tilde{u}(s)) ds,$$

(SC5a)

$$[\text{or } \langle \xi, f(x_t, \tilde{u}_t) \rangle \leq -L(x_t, \tilde{u}_t), \forall \xi \in \partial P W(x_t)]$$

and

$$x(s; t, x_t, \tilde{u}) \in S, \quad s \in [t, t + \delta).$$

(SC5b)

$$[\text{or } \langle \xi, f(x_t, \tilde{u}_t) \rangle \leq 0, \forall \xi \in N^P_S(x_t)]$$

Then, the MPC strategy is stabilizing (i.e. for a sufficiently small inter-sample time $\delta$ we have $\|x^*(t)\| \to 0$ as $t \to +\infty$).
Example of a simple nonlinear system

Consider the system

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -x_1(t) + 2x_1(t)u(t),
\end{align*}
\]

It cannot be driven to the origin in finite time. Linearizing around the origin we get

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x,
\]

poles at \{-j, j\}

The linearization is not stabilizable.

Consider \( S := \{(x_1, x_2) : x_1 = -x_2\} \)

If \( u = 0 \) the trajectories are

\[
\begin{align*}
x_1(t) &= x_1(0) \sin(t) \\
x_2(t) &= x_2(0) \cos(t),
\end{align*}
\]

\( S \) can be reached in \( T \geq 2\pi \).

Choose \( L(x) = \|x\|^2 \) and \( W(x) = \|x\|^2 \); SC is satisfied since

\[
\dot{W} = 2x^T \cdot f(x, 1) = -4x_1^2 \leq -L(x) = -2x_1^2.
\]

MPC is stabilizing.
Example: A car-like system
Consider the system

\[
\begin{align*}
\dot{x} &= v \cdot \cos \theta \\
\dot{y} &= v \cdot \sin \theta \\
\dot{\theta} &= v \cdot c
\end{align*}
\]

with control inputs $v$ and $c$ satisfying

$v \in [0, v_{\text{max}}]$ and $c \in [-c_{\text{max}}, c_{\text{max}}]$.

Stability achieved choosing $S$ as in picture

$T = 2\pi R_{\text{min}} / v_{\text{max}}$.

$L(x, y, \theta) := x^2 + y^2 + \theta^2$,

$W(x_0, y_0, \theta_0) := \int_0^{\bar{t}} L(x(t), y(t), \theta(t)) \, dt$

where $\bar{t}$ is the time to reach the origin.

SC is satisfied. MPC is stable.
Further details:

Fontes, F.A.C.C.:  
A general framework to design stabilizing nonlinear model predictive controllers.  
Systems & Control Letters 42 (2001) 127–143

F. A. C. C. Fontes.  
Optimisation-Based Control of Constrained Nonlinear Systems,  
Imperial College of Science Technology and Medicine, University of London,  

Mayne, D. Q., Rawlings, J. B., Rao, C. V., Scokaert, P. O.  
Constrained model predictive control: Stability and optimality.  
Automatica, 36(6),(2000), 789-814.

Grüne, Lars.  
"Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems.”  
Phenomena best studied in continuous-time

Discontinuous Feedbacks

Stability

Discontinuous Feedbacks

Bang-bang control

Path-following

Impulsive systems
Nonholonomic Systems

Are completely controllable but instantaneously cannot move in certain directions (Ex: wheeled vehicles, robot manipulators, rolling sphere, ...)

Example: A car-like system

\[
\begin{align*}
\dot{x} &= v \cdot \cos \theta \\
\dot{y} &= v \cdot \sin \theta \\
\dot{\theta} &= v \cdot c
\end{align*}
\]

with control inputs \( v \) and \( c \) satisfying

\( v \in [0, v_{\text{max}}] \) and \( c \in [-c_{\text{max}}, c_{\text{max}}] \).

(Minimum turning radius \( R_{\text{min}} = c_{\text{max}}^{-1} \))

- Nonholonomic constraint: velocity vector is orthogonal to the wheel axle
  \((\dot{x}(t), \dot{y}(t))^T (\sin \theta(t), -\cos \theta(t)) = 0\)
- Linearizing around any point \((x_0, y_0, \theta_0, v = 0)\) results in an uncontrollable system.
- Cannot be stabilized by a continuous (time-invariant) feedback.
Cannot be stabilized by a continuous feedback.

Suppose we want to go to $x = y = \theta = 0$ and we are at $x = 0$ and $\theta = \pi$.

- If $y \geq 2R_{\min}$ ⇒ turn left.
- If $y < 2R_{\min}$ ⇒ turn right.

which is a discontinuous feedback.
Cannot be stabilized by a continuous feedback.

Suppose we want to go to $x = y = \theta = 0$ and we are at $x = 0$ and $\theta = \pi$.
- If $y \geq 2R_{\text{min}} \Rightarrow$ turn left.
- If $y < 2R_{\text{min}} \Rightarrow$ turn right.
which is a discontinuous feedback.

More precisely ...

**Theorem (Brockett 83)**

If the system $\dot{x} = f(x, u)$, with $u \in U$, admits a continuous stabilizing feedback, then for any neighbourhood $\mathcal{X}$ of zero, the set $f(\mathcal{X}, U)$ contains a neighbourhood of zero.

In the car example, if
$\mathcal{X} := \{(x, y, \theta) : |x| < 1, |y| < 1, |\theta| < \pi/6\}$
then
$f(\mathcal{X}, U)$ has no points of the form $(\dot{x}, \dot{y}, \dot{\theta})$ with $\dot{x} < 0$. 
Discontinuous Feedback: Difficulties

- The trajectory $x$, solution to

$$\dot{x}(t) = f(x(t), k(x(t))), \quad x(0) = x_0$$

is only defined, in classical (Caratheodory) sense, if (among other requirements) if the right-hand side is a locally Lipschitz continuous function of $x$.

- Using Filippov solution-concept some controllable systems cannot be stabilized. (Ryan 94, Coron&Rosier 94).

- Problem solved if we use “sampling-feedback” solution concept (Clarke, Ledyaev, Sontag, Subbotin 97)
Phenomena best studied in continuous-time

Discontinuous Feedbacks: (Un)definition of trajectory

- A Runner, a Cyclist and a Dog depart from the same point, at the same time, with the same direction, with velocities $v$, $2v$, and $3v$.

- Runner and Cyclist keep the same direction
  Dog uses a "discontinuous strategy": goes back and forward between cyclist and runner

Where is the dog after a certain time $T$?
Discontinuous Feedbacks: (Un)definition of trajectory

- The problem is solved if the dog keeps the decision for some time $\delta > 0$
Discontinuous Feedback: “sampling-feedback” solution

(Clarke, Ledyaev, Sontag, Subbotin IEEE TAC97)

- Take a sequence of sampling instants \( \pi := \{t_i\}_{i \geq 0} \) in \([0, \infty)\) with \( t_0 < t_1 < t_2 < \ldots \)

\[
\dot{x}(t) = f(x(t), k(x([t]_\pi))), \quad x(0) = x_0
\]

where \([t]_\pi := \max_i \{t_i \in \pi : t_i \leq t\}\)

i.e. The feedback is not a function of the state at every instant of time, rather it is a function of the state at the last sampling instant.

This solution concept combines naturally with our MPC sampled-data framework.

Just use \( \delta > 0 \), as you’d naturally do, and IT WORKS!

What about in discrete-time systems?
Our MPC framework

Consider a sequence $\{t_i\} \geq 0$ s.t. $t_{i+1} = t_i + \delta$, $\delta > 0$:

1. Measure state of the plant $x_{t_i}$
2. Get $\bar{u} : [t_i, t_i + T] \mapsto \mathbb{R}^m$ solution to the OCP:

   Minimize $\int_0^{t_i+T} L(t, x(t), u(t)) dt + W(x(t_i + T))$

   subject to $\dot{x}(t) = f(t, x(t), u(t))$ a.e. $t \in [t_i, t_i + T]$
   $x(t_i) = x_{t_i}$
   $u(t) \in U(t)$ a.e. $t \in [t_i, t_i + T]$
   $x(t) \in X$ a.e. $t \in [t_i, t_i + T]$
   $x(t_i + T) \in S$

3. Apply to the plant the control $u^*(t) := \bar{u}(t)$ in the interval $[t_i, t_i + \delta]$. (the remaining control $\bar{u}(t)$, $t > t_i + \delta$ is discarded)
4. Repeat for the next sampling time $t_i = t_i + \delta$
Consider the system

\[
\begin{align*}
\dot{x}(t) &= (u_1(t) + u_2(t)) \cdot \cos \theta(t) \\
\dot{y}(t) &= (u_1(t) + u_2(t)) \cdot \sin \theta(t) \\
\dot{\theta} &= (u_1(t) - u_2(t)).
\end{align*}
\]

with \( \theta(t) \in [-\pi, \pi] \), and the controls \( u_1, u_2(t) \in [-1, 1] \).

Stability achieved choosing:

\( S \) to be the set of points s.t. the heading angle \( \theta \)

is pointing towards the origin of the plane \((x, y) = (0, 0)\).

\[ T = \pi/2. \]

\[ L(x, y, \theta) := x^2 + y^2 + \theta^2, \]

\[ W(x_0, y_0, \theta_0) := \int_0^{\bar{t}} L(x(t), y(t), \theta(t)) dt \]

where \( \bar{t} \) is the time to reach the origin.

SC is satisfied. MPC is stable.
Further details:

Fontes, F.A.C.C. (2003):
Discontinuous feedbacks, discontinuous optimal controls, and continuous-time model predictive control.

Asymptotic controllability implies feedback stabilization.
IEEE Transactions on Automatic Control, 42(10), 1394-1407.
Phenomena best studied in continuous-time

- Bang-bang control

Stability

Discontinuous Feedbacks

Bang-bang control

Path-following

Impulsive systems
A Robust MPC approach

system with bounded disturbances

\[ \dot{x}(t) = f(x(t), u(t), d(t)) \quad \text{a.e. } t \geq 0, \quad x(0) = x_0 \]

\[ u(t) \in [u^{\text{min}}, u^{\text{max}}], \quad d(t) \in [d^{\text{min}}, d^{\text{max}}] \]

**Main challenge:** Solve min-max OCP searching for the best feedback strategy under the worst possible disturbance.

**Approach:** Restrict to feedbacks defined by a small set of parameters: e.g. bang-bang feedbacks defined by switching surface.

**bang-bang feedback**

\[ k_j^\Lambda(x) = \begin{cases} 
0 & \text{if } x \in \Theta, \\
u_j^{\text{max}} & \text{if } \sigma_j^\Lambda(x) \geq 0, \\
u_j^{\text{min}} & \text{if } \sigma_j^\Lambda(x) < 0, 
\end{cases} \]

where \( s^\Lambda(x) = 0 \) is a switching surface defined by a parameter matrix \( \Lambda \).
A Robust MPC approach (cont.)

\[ P(x_t, T_c, T_p): \min_{\Lambda_1, \ldots, \Lambda_N} \max_{d \in D(0, T_p)} \int_t^{t+T_p} L(x(s), u(s)) ds + W(x(t + T_p)) \]  
subject to:

\[ x(t) = x_t \]
\[ \dot{x}(s) = f(x(s), u(s), d(s)) \quad \text{a.e. } s \in [t, t + T_p] \]
\[ x(s) \in X \quad \text{for all } s \in [t, t + T_p] \]
\[ x(t + T_p) \in S. \]

where

\[ u(s) = k^{\Lambda_i}(x(\lfloor s \rfloor)) \quad s \in [t + (i - 1)\delta, t + i\delta), \quad i = 1, \ldots, N_p \]
Nonholonomic vehicle model

Our vehicle: A differential-drive mobile robot

\[
\begin{align*}
\dot{x}(t) &= (1 + d(t))(u_1(t) + u_2(t)) \cdot \cos \theta(t) \\
\dot{y}(t) &= (1 + d(t))(u_1(t) + u_2(t)) \cdot \sin \theta(t) \\
\dot{\theta} &= (1 + d(t))(u_1(t) - u_2(t)).
\end{align*}
\]

with constrained controls \(u_1, u_2(t) \in [-1, 1]\) and perturbations \(d(t) \in [-1/4, 1/4]\).
Trajectory controller: Stability

A set of design parameters that guarantees stability is the following [FontesMagni03]:

\[ L(x, y, \theta) = x^2 + y^2 + \theta^2, \]

\[ W(x, y, \theta) = \frac{1}{3}(r^3 + |\theta|^3) + r\theta^2 \quad \text{with} \quad r = \sqrt{x^2 + y^2}. \]

\[ T_p = \frac{2\pi}{3} \]

\[ S := \{(x, y, \theta) \in \mathbb{R}^2 \times [-\pi, \pi] : \phi_m(x, y) \leq \theta \leq \phi_M(x, y) \}
\quad \forall (x, y, \theta) \in \Theta \quad \forall (x, y) = (0, 0). \]

Could this be done with Discrete-time models?
Further details:

F. A. C. C. Fontes and L. Magni.
Min-max model predictive control of nonlinear systems using discontinuous feedbacks.

F. A. C. C. Fontes and L. Magni.
A Generalization of Barbalat’s Lemma with Applications to Robust Model Predictive Control.

Fontes, F.A.C.C., Magni, L., Gyurkovics, E.:
Sampled-data model predictive control for nonlinear time-varying systems: Stability and robustness.
Phenomena best studied in continuous-time

- Path-following

Stability

Discontinuous Feedbacks

Bang-bang control

Path-following

Impulsive systems
Phenomena best studied in continuous-time

Path-following

Problem Objective

What do we mean by “to control”?

- point-to-point motion (what we’ve seen so far)
- trajectory tracking (much easier)
- path-following (one more degree of freedom)
- accomplish mission objective (why not explore further degrees of freedom?)

**Trajectory tracking**: The aim is to follow, as close as possible, a trajectory in state space (i.e., a geometric path in the cartesian space together with an associated timing law) starting from a given initial configuration.

**Path following**: The robot must reach and follow a geometric path in the cartesian space starting from a given initial configuration, but no associated timing law is specified.
Path-following

Given a path parameterized by \( \tau \mapsto (x_d(\tau), y_d(\tau)) \) for \( \tau \in [0, T_d] \):

1. Find an initial point \( Q_0 \) in the path to start the trajectory of the virtual reference vehicle. Given the initial (current) position of our vehicle \((x_0, y_0)\), find the path parameter value that corresponds to a point in the path that is the closest to \((x_0, y_0)\).

\[
\tau_0 = \arg\min_{\tau \in [0, T_d]} \| (x_d(\tau), y_d(\tau)) - (x_0, y_0) \|
\]

2. Select a speed profile at which the path is to be followed

\[
\nu : [0, T] \mapsto \mathbb{R}_+
\]

3. Find a feedback control to track the trajectory of the virtual reference vehicle

\[
t \mapsto (x_r(t), y_r(t))
\]
Path-following with MPC

Minimize over $\nu \in \mathbb{R}_+$ and $u$

$$
\int_0^{T_d/\nu} \left( \|q(t) - q_d(\tau(t))\|_Q^2 + \|u(t)\|_R^2 \right) dt + \|q(T_d/\nu)\|_P^2,
$$

subject to:

$$
\begin{align*}
\dot{q}(t) &= f(t, q(t), u(t)) & \text{a.e. } t \in [0, T_d/\nu] \\
q(0) &= q_{t_0} \\
\tau_0 &= \operatorname{argmin}_{\tau \in [0, T_d]} \|q_d(\tau) - q_{t_0}\| \\
\tau(t) &= \tau_0 + \nu t \\
u(t) &\in U & \forall t \in [0, T_d/\nu].
\end{align*}
$$

where the path is parameterized by

$$
t \mapsto q_d(\tau), \quad \tau \in [0, T_d]
$$
Path-following example

Consider a differential-drive robot to follow a eight-shaped path

\[ x_d(\tau) = \sin(\tau/10), \quad \tau \in [0, 38\pi] \]
\[ y_d(\tau) = \sin(\tau/20), \quad \tau \in [0, 38\pi] \]

By selecting \( \nu \) and an initial point in the path we define the trajectory for a virtual reference vehicle

\[ x_r(t) = \sin(\tau(t)/10) \]
\[ y_r(\tau) = \sin(\tau(t)/20) \]
\[ \tau(t) = \tau_0 + \int_0^t \nu(s)ds. \]

and by inverse kinematics we obtain the control of the reference vehicle

\[ \theta_r(t) = \text{ATAN2} \left( \dot{y}_r, \dot{x}_r \right) \]
\[ \nu_r(t) = \pm \sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \]
\[ w_r(t) = \frac{\ddot{y}_d(t) \dot{x}_d(t) - \dot{x}_d(t) \ddot{y}_d(t)}{\dot{y}_d^2(t) + \dot{x}_d^2(t)}, \quad \text{a.e.} \]
Phenomena best studied in continuous-time

Path-following

Path-following example

Apply the MPC with a linearized model:

Minimize

$$\int_{0}^{T_{d}/\nu} \left( \| q(t) - q_{d}(\tau(t)) \|_{Q}^{2} + \| u(t) \|_{R}^{2} \right) dt + \| q(T_{d}/\nu) \|_{P}^{2},$$

subject to:

$$\dot{e}(t) = Ae(t) + Bu(t) \quad a.e.t \in [0, T_{d}/\nu]$$

$$e(0) = R_{\theta}(q_{d}(\tau_{0}) - q(t_{i}))$$

$$\tau_{0} = \arg\min_{\tau \in [0, T_{d}]} \| q_{d}(\tau) - q_{t_{i}} \|$$

$$\tau(t) = \tau_{0} + \nu t \quad \forall t \in [0, T_{d}/\nu]$$

$$|u_{1}(t)| \leq U_{\max} \quad \forall t \in [0, T_{d}/\nu]$$

$$|u_{2}(t)| \leq U_{\max} \quad \forall t \in [0, T_{d}/\nu].$$

where $Q$ and $R$ are chosen to be p.d., and $P$ solves the Riccati equation

$$A'P + PA - PBR^{-1}B'P + Q = 0,$$

 guaranteeing stability.
Path-following for aerial vehicles

Linearization with varying speed profile I

We would like to compare our trajectory with the reference trajectory in a discrete set of instants of time when the reference trajectory is followed at varying speed. To do that, we consider non-equidistant time instants \( \{t_k\}_{k \geq 0} \) such that

\[
t_{k+1} = t_k + \alpha(k) T, \quad k = 0, 1, \ldots, \quad \alpha(k) \in (0, 1],
\]

where \( T \) is the nominal sampling period. Discretizing with Euler explicit method, we have

\[
\xi(t_{k+1}) = \xi(t_k) + \alpha(k) T f(\xi(t_k), u(t_k)), \quad k = 0, 1, \ldots
\]

To linearize, we redefine the collection of linearization points along the reference trajectory to be

\[
\mathbb{L} \triangleq \{ \dot{l}^j = (\xi^j, u^j, \alpha^j), \quad j = 1, 2, \ldots, N_l \},
\]

with one additional component \( \alpha^j \) which is chosen to be \( \alpha^j = 1 \), corresponding to the reference trajectory at nominal speed. The linearization around a point \( \dot{l}^j \in \mathbb{L} \) yields

\[
f(\xi(t_k), u(t_k)) \approx f(\xi^j, u^j) + f_x(\xi^j, u^j)(\xi(t_k) - \xi^j) + f_u(\xi^j, u^j)(u(t_k) - u^j)
\]
Phenomena best studied in continuous-time

Path-following

Linearization with varying speed profile II

and

$$\xi(t_{k+1}) \simeq \xi(t_k) + \alpha(k) T f_x(\xi^i, u^i) \xi(t_k) + \alpha(k) T f_u(\xi^i, u^i) u(t_k)$$

$$+ \alpha(k) T (f(\xi^i, u^i) - f_x(\xi^i, u^i) \xi^i - f_u(\xi^i, u^i) u^i) \tag{8}$$

$$\simeq (I + \alpha^i T f_x(\xi^i, u^i)) \xi(t_k) + \alpha^i T f_u(\xi^i, u^i) u(t_k)$$

$$+ \alpha(k) T (f(\xi^i, u^i) - f_x(\xi^i, u^i) \xi^i - f_u(\xi^i, u^i) u^i). \tag{9}$$

We note that (8) is a bilinear model and a further approximation ($\alpha(k) = \alpha^i$ in the second and third terms) leads to the linear model (9).

Denoting $\xi(t_k)$ and $u(t_k)$ by $\xi(k)$ and $u(k)$ to simplify notation, we obtain the discrete-time linearized model

$$\xi(k + 1) = A_j \xi(k) + B^1_j \alpha(k) + B^2_j u(k),$$

where

$$A_j = I + \alpha^j T f_x(\xi^i, u^i),$$

$$B^1_j = T (f(\xi^i, u^i) - f_x(\xi^i, u^i) \xi^i - f_u(\xi^i, u^i) u^i),$$

$$B^2_j = \alpha^j T f_u(\xi^i, u^i).$$
Further details


Phenomena best studied in continuous-time

Impulsive systems

Phenomena best studied in continuous-time

Stability

Discontinuous Feedbacks

Bang-bang control

Path-following

Impulsive systems
Impulsive dynamical Systems (IS) are systems in which the state trajectories can have discontinuities (jumps, sudden changes) in response to (impulsive) controls.

Examples

- mechanical systems subject to collision: the velocity has a discontinuity during collision (e.g. walking robots),
- hybrid systems (e.g. in a car, engine revolutions have a discontinuity during a gear change)
- singular control action (e.g. a space rocket when releasing a stage has a discontinuity in the mass)
- reposition of a stock of a product in inventory control, investment policies in economic systems, ...

\[ u \] conventional control
\[ \mu \] impulsive control
How to model IS?

- Impulsive control is often an instantaneous action with high impact on the results.
- The timing of this action is often crucial.
  \[ \Rightarrow \] Time instants of the impulsive action must be modelled appropriately.

- Discrete-time models?
- Continuous-time models with \textit{a priori} knowledge of instants of the impulsive action?
- Continuous-time models without \textit{a priori} knowledge of instants of the impulsive action?
A simple example with discontinuous trajectory: Billiard collision

Initial positions: \((x_1, y_1) = (0, 0), \quad (x_2, y_2) = (1 + 2r, 1)\);
Initial speed: \(|\dot{v}_1| = 1, \quad |\dot{v}_1| = 0|;
Collision instant: \(T = \sqrt{2}\).
Collision condition: \(|(x_1, y_1) - (x_2, y_2)| \leq 2r|.
key feature in IS

A **key feature** for sampled-data models when trajectory has discontinuities:

**The discontinuity instants should be on (or very near) a sampling instant**

**But** discontinuity instants might not be known in advance. They might depend on a chosen control action!

**MPC can help here!**
Main idea: why MPC for IS?

Main idea

*MPC can plan/predict not only future control moves, but also plan/predict the future sampling instants sequence.*

... and we provide conditions for stability of MPC for IS.
Phenomena best studied in continuous-time

Impulsive systems

Some references

Model predictive control of impulsive dynamical systems.
*IFAC Proceedings NMPC 2012*

Hamilton-jacobi-bellman equation and feedback synthesis for impulsive control.
*IEEE Transactions on Automatic Control, 57*(1), 244–249.

L Pereira, FACC Fontes, AP Aguiar, JB Sousa.
An Optimization-Based Framework for Impulsive Control Systems
In *Developments in Model-Based Optimization and Control*, S. Olaru et al (eds.),
Part III

Efficient solution of Continuous-Time optimal control problems
Efficient solution of Continuous-Time optimal control problems

Numerical Solution of OCP

Mesh Refinement for Optimal Control

Mesh refinement in MPC
Numerical Solution of Optimal Control Problems

To solve an OCP isn’t it easier to use a discrete-time model from the beginning?

- Yes, if you’re planning to code the solver from scratch.
- No, if you use one of the plenty available solvers for CT OCP (ICLOCS, PRONTO, BOCOOP, ACADO, CasADi, OCPID-DAE1, gPOPS, ...)

Isn’t it faster to obtain solution with DT models?

- NO!
Consider the OCP to be solved numerically

\[
\begin{align*}
\text{Minimize} & \quad \int_{t_0}^{t_f} L(t, x(t), u(t)) \, dt + g(t_f, x(t_f)) \\
\text{subject to} & \quad \dot{x}(t) = f(t, x(t), u(t)) \quad \text{a.e. } t \in [t_0, t_f] \\
& \quad u \in U(t) \subset \mathbb{R}^m \quad \text{a.e. } t \in [t_0, t_f] \\
& \quad h(t, x(t)) \leq 0 \quad \forall t \in [t_0, t_f] \\
& \quad x(t_0) \in X_0 \quad \text{and} \quad x(t_f) \in X_1
\end{align*}
\]
Efficient solution of Continuous-Time optimal control problems

Numerical Solution of OCP

Discretizations errors

Consider the OCP to be solved numerically

\[
\text{Minimize } \int_{t_0}^{t_f} L(t, x(t), u(t)) \, dt + g(t_f, x(t_f)) \quad \rightarrow \text{running \& final costs}
\]

subject to

\[
\begin{align*}
\dot{x}(t) &= f(t, x(t), u(t)) \quad \text{a.e. } t \in [t_0, t_f] \quad \rightarrow \text{dynamic constraints} \\
u &\in U(t) \subset \mathbb{R}^m \quad \text{a.e. } t \in [t_0, t_f] \quad \rightarrow \text{input constraints} \\
h(t, x(t)) &\leq 0 \quad \forall t \in [t_0, t_f] \quad \rightarrow \text{path constraints} \\
x(t_0) &\in X_0 \quad \text{and} \quad x(t_f) \in X_1 \quad \rightarrow \text{boundary conditions}
\end{align*}
\]

Select a time grid

\[
\pi := \{t_i\}_{i \geq 0}
\]

in \([t_0, t_f]\) with \(t_{i+1} = t_i + \delta_i\), with \(\delta_i > 0\).

The dynamics constraint

\[
\dot{x}(t) = f(t, x(t), u(t)) \quad \text{a.e. } t \in [t_0, t_f]
\]

has a discrete approximation in \(\pi\) (e.g. using Euler)

\[
x(t_{i+1}) = x_{ti} + \delta_i f(t, x(t), u(t)) \quad t_i \in \pi
\]
Discretizations errors

Select a time grid

\[ \pi := \{ t_i \}_{i \geq 0} \]

in \([t_0, t_f]\) with \(t_{i+1} = t_i + \delta_i\), with \(\delta_i > 0\).

The dynamics constraint

\[ \dot{x}(t) = f(t, x(t), u(t)) \text{ a.e. } t \in [t_0, t_f] \]

has a discrete approximation in \(\pi\) (e.g. using Euler)

\[ x(t_{i+1}) = x(t_i) + \delta_i f(t, x(t), u(t)) \quad t_i \in \pi \]

Let \(\epsilon_\pi\) be an estimate for the discretization error.

- The discretization procedures are always based on some form of local linearization (though they might have higher order terms).
- So, we should expect non-negligible errors for highly nonlinear systems (such as NH systems).
Motivation: Selecting a Mesh for Direct Methods
Motivation: Selecting a Mesh for Direct Methods
Motivation: Selecting a Mesh for Direct Methods

COARSE Mesh

ADAPTED Mesh

FINE Mesh

INACCURATE

FAST
Motivation: Selecting a Mesh for Direct Methods

- COARSE Mesh
- ADAPTED Mesh
- FINE Mesh

- INACCURATE
- FAST
- ACCURATE
- SLOW
Motivation: Selecting a Mesh for Direct Methods

COARSE Mesh

ADAPTED Mesh

FINE Mesh

INACCURATE

ACCURATE

FAST

SLOW
Time-grid selection vs. Time-grid refinement

Time-grid selection:
Time-grid selection vs. Time-grid refinement

Time-grid selection:
No (clever) ideia of how to do it (a priori)
Time-grid selection vs. Time-grid refinement

**Time-grid selection:**
No (clever) idea of how to do it (a priori)

A sensible heuristic: reduce $\delta_j$ when the system has a higher nonlinear behavior (in the car-like, when curvature is higher).
Time-grid selection vs. Time-grid refinement

**Time-grid selection:**
No (clever) idea of how to do it (a priori)

A sensible heuristic: reduce $\delta_i$ when the system has a higher nonlinear behavior (in the car-like, when curvature is higher).
But how do we know these points in advance?
Time-grid selection vs. Time-grid refinement

**Time-grid selection:**
No (clever) idea of how to do it (a priori)

A sensible heuristic: reduce $\delta_i$ when the system has a higher nonlinear behavior (in the car-like, when curvature is higher).
But how do we know these points in advance?

**Solution:**
Refinement.

Start with a coarse grid and refine it adaptively
Efficient solution of Continuous-Time optimal control problems

Mesh Refinement for Optimal Control

Numerical Solution of OCP

Mesh Refinement for Optimal Control

Mesh refinement in MPC
Time–mesh refinement

Time–mesh refinement

- is not a new idea;
- is widely studied in the context of PDE.

Contributions (in optimal control)

- [Betts 1998], [Zhao 2011], [Patterson, Hager 2014], …

Our refinement strategy

- **Basis**: block–structured adaptive mesh refinement (widely used in fluid mechanics);
- **Multi–level**: different levels of refinement in a single iteration;
- **Adjoint Multipliers**: information used to refine.
Time–mesh refinement

- is not a new idea;
- is widely studied in the context of PDE.

Contributions (in optimal control)

- [Betts 1998], [Zhao 2011], [Patterson, Hager 2014], ...

Our refinement strategy

- Basis: block–structured adaptive mesh refinement (widely used in fluid mechanics);
- Multi–level: different levels of refinement in a single iteration;
- Adjoint Multipliers: information used to refine.
Time–mesh refinement

- is not a new idea;
- is widely studied in the context of PDE.

Contributions (in optimal control)
- [Betts 1998], [Zhao 2011], [Patterson, Hager 2014], ... 

Our refinement strategy
- **Basis**: block–structured adaptive mesh refinement (widely used in fluid mechanics);
- **Multi–level**: different levels of refinement in a single iteration;
- **Adjoint Multipliers**: information used to refine.
Multi–Level Time–Mesh Refinement Algorithm

**Data**: Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result**: optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

while stopping criteria not met do
  apply the discretisation scheme according to the multi–level refinement criteria;
  transcribe the OCP;
  create warm–start;
solve NLP;
estimate discretisation error;
end
Multi–Level Time–Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;

select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

while stopping criteria not met do
  apply the discretisation scheme according to the multi–level refinement criteria;
  transcribe the OCP;
  create warm–start;
  solve NLP;
  estimate discretisation error;
end
Multi-Level Time-Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;
select (coarse) time-mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

**while stopping criteria not met do**

- apply the discretisation scheme according to the multi-level refinement criteria;
- transcribe the OCP;
- create warm-start;
- solve NLP;
- estimate discretisation error;

**end**
Efficient solution of Continuous-Time optimal control problems

Mesh Refinement for Optimal Control

Multi–Level Time–Mesh Refinement Algorithm

**Data**: Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result**: optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

**while stopping criteria not met do**

- apply the discretisation scheme according to the multi–level refinement criteria;
- transcribe the OCP;
- create warm–start;
- solve NLP;
estimate discretisation error;

**end**
Multi-Level Time-Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;
select (coarse) time-mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

**while stopping criteria not met do**
apply the discretisation scheme according to the **multi-level** refinement criteria;
transcribe the OCP;
create warm-start;
solve NLP;
estimate discretisation error;

**end**
Multi–Level Time–Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

**while stopping criteria not met do**
apply the discretisation scheme according to the **multi–level** refinement criteria;
transcribe the OCP;
create warm–start;
solve NLP;
estimate discretisation error;

**end**
Efficient solution of Continuous-Time optimal control problems

Mesh Refinement for Optimal Control

Multi–Level Time–Mesh Refinement Algorithm

Data: Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

Result: optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

while stopping criteria not met do
apply the discretisation scheme according to the multi–level refinement criteria;
transcribe the OCP;
create warm–start;
solve NLP;
estimate discretisation error;
end
Multi–Level Time–Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

while stopping criteria not met do
apply the discretisation scheme according to the **multi–level** refinement criteria;
transcribe the OCP;
create warm–start;
solve NLP;
estimate discretisation error;
end
Multi–Level Time–Mesh Refinement Algorithm

**Data:** Cost functionals, dynamics, constraints, initial/terminal boundaries, parameters

**Result:** optimal trajectories, controls, error initialisation;
select (coarse) time–mesh;
discretize and transcribe the OCP;
solve NLP;
estimate discretisation error;

**while stopping criteria not met do**
  apply the discretisation scheme according to the **multi–level** refinement criteria;
  transcribe the OCP;
  create warm–start;
  solve NLP;
  estimate discretisation error;

**end**
Refinement and Stopping criteria

Refinement criteria

- local error on $x$ (trajectory)
- local error on $q$ (adjoint multipliers)

Why?

- $q$ provides sensitivity information
- $q$ is solution of a linear ODE system (efficiently solved with high accuracy)

- both simultaneously

Stopping criterion

- local error on $x$ (trajectory)
Refinement and Stopping criteria

Refinement criteria

- local error on $x$ (trajectory)
- local error on $q$ (adjoint multipliers)

Why?

- $q$ provides sensitivity information
- $q$ is solution of a linear ODE system (efficiently solved with high accuracy)
- both simultaneously

Stopping criterion

- local error on $x$ (trajectory)
Efficient solution of Continuous-Time optimal control problems

Mesh Refinement for Optimal Control

Adjoint Multipliers

Error on the adjoint multipliers

\[-\dot{p}(t) = -q_x - \lambda L_x \quad \text{a.e.} \quad t \in [t_0, t_f]\]

\[q(t) = \begin{cases} 
    p(t) + \int_{[t_0, t]} h_x(x^*(s))d\mu(s) & t \in [t_0, t) \\
    p(t_f) + \int_{[t_0, t_f]} h_x(x^*(s))d\mu(s) & t = t_f
\end{cases}\]

\[\epsilon_q = |q_{\text{IPOPT}} - q_{\text{MP}}|\]
Car–Like system – Minimum Energy

Example using mesh refinement strategy

Minimize \( \int_{0}^{10} u^2(t) dt \)

subject to \( \dot{x} = u \cos(\psi) \quad \forall \ t \in [0, 10] \)
\( \dot{y} = u \sin(\psi) \quad \forall \ t \in [0, 10] \)
\( \dot{\psi} = u \ c \quad \forall \ t \in [0, 10] \)

\( 0 \leq u(t) \leq 1 \quad \forall \ t \in [0, 10] \)
\( -0.7 \leq c(t) \leq 0.7 \quad \forall \ t \in [0, 10] \)

\( 10(x - 5)^2 \leq y - 1 \quad \forall \ t \in [0, 10] \)

\( (x_n - x_f)^2 + (y_n - y_f)^2 + (\theta_n - \theta_f)^2 \leq 0.1 \quad \forall \ t \in [0, 10] \)

\( x_0 = (x_0, y_0, \psi_0) = (0, 0, 0) \)
\( x_f = (x_f, y_f, \psi_f) = (10, 0, 0) \)
Car–Like system – Minimum Energy

Example using mesh refinement strategy

Minimize \( \int_0^{10} u^2(t)dt \)

subject to \( \dot{x} = u \cos(\psi) \) \( \forall t \in [0, 10] \)
\( \dot{y} = u \sin(\psi) \) \( \forall t \in [0, 10] \)
\( \dot{\psi} = u c \) \( \forall t \in [0, 10] \)

\( 0 \leq u(t) \leq 1 \) \( \forall t \in [0, 10] \)
\( -0.7 \leq c(t) \leq 0.7 \) \( \forall t \in [0, 10] \)
\( 10(x - 5)^2 \leq y - 1 \) \( \forall t \in [0, 10] \)
\( (x_n - x_f)^2 + (y_n - y_f)^2 + (\theta_n - \theta_f)^2 \leq 0.1 \) \( \forall t \in [0, 10] \)

\( x_0 = (x_0, y_0, \psi_0) = (0, 0, 0) \)
\( x_f = (x_f, y_f, \psi_f) = (10, 0, 0) \)
Time mesh refinement

Car–like System: Trajectory and controls

**Figure:** Trajectory and Controls
Time mesh refinement

Car–like System: Error evolution

Figure: Error evolution (in logarithmic scale)
Multi–Level Adaptive Time–Mesh Refinement

Defining

- $\pi_{ML}$: time–mesh given by the multi-level adaptive refinement strategy
- $\pi_F$: time–mesh with equidistant–spacing considering the smallest time step of $\pi_{ML}$

Considering

- Stopping criterion: $||\varepsilon_x||_\infty \leq \varepsilon_x^{\text{max}} = 5E^{-5}$
- Refinement criterion: $\varepsilon_q(t) \leq \varepsilon_q^{\text{max}} = 5E^{-5}$
- Refinement levels: $\bar{\varepsilon}_q = (5E^{-5}, 5E^{-4}, 5E^{-3}, 5E^{-2}, 5E^{-1})$
Multi-Level Adaptive Time-Mesh Refinement

Defining
- $\pi_{ML}$ time-mesh given by the multi-level adaptive refinement strategy
- $\pi_F$ time-mesh with equidistant-spacing considering the smallest time step of $\pi_{ML}$

Considering
- Stopping criterion: $||\varepsilon_x||_\infty \leq \varepsilon_x^{\text{max}} = 5E^{-5}$
- Refinement criterion: $\varepsilon_q(t) \leq \varepsilon_q^{\text{max}} = 5E^{-5}$
- Refinement levels: $\bar{\varepsilon}_q = (5E^{-5}, 5E^{-4}, 5E^{-3}, 5E^{-2}, 5E^{-1})$
Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>$\pi_j$</th>
<th>$N_j$</th>
<th>$\Delta t_j$</th>
<th>$l_j$</th>
<th>Objective</th>
<th>$|\varepsilon^{(j)}<em>x|</em>\infty$</th>
<th>$|\varepsilon^{(j)}<em>q|</em>\infty$</th>
<th>CPU time (s)</th>
<th>Solve</th>
<th>$\varepsilon_x$</th>
<th>$\varepsilon_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_0$</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E$^{-2}$</td>
<td>2.443E$^{-2}$</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E$^{-4}$</td>
<td>1.444E$^{-3}$</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
<td></td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
<td></td>
</tr>
<tr>
<td>$\pi_{ML}$</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
<td></td>
</tr>
<tr>
<td>$\pi_F$</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td>9.7805377</td>
<td>3.996E$^{-5}$</td>
<td>$-$</td>
<td>4840.185</td>
<td>773.192</td>
<td>$-$</td>
<td></td>
</tr>
</tbody>
</table>

Main remarks

- Local errors of the high order of magnitude
Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>$\pi_j$</th>
<th>$N_j$</th>
<th>$\Delta t_j$</th>
<th>$I_j$</th>
<th>Objective</th>
<th>$|\varepsilon_x^{(j)}|_\infty$</th>
<th>$|\varepsilon_q^{(j)}|_\infty$</th>
<th>CPU time (s)</th>
<th>Solve</th>
<th>$\varepsilon_x$</th>
<th>$\varepsilon_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_0$</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E$^{-2}$</td>
<td>2.443E$^{-2}$</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E$^{-4}$</td>
<td>1.444E$^{-3}$</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
<td></td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
<td></td>
</tr>
<tr>
<td>$\pi_{ML}$</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
<td></td>
</tr>
<tr>
<td>$\pi_F$</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td>9.7805377</td>
<td>3.996E$^{-5}$</td>
<td>–</td>
<td>4840.185</td>
<td>773.192</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

Main remarks

- $\pi_{ML}$ has only 32.4% of the nodes of $\pi_F$
- Local errors of the high order of magnitude
## Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>$\pi_j$</th>
<th>$N_j$</th>
<th>$\Delta t_j$</th>
<th>$l_j$</th>
<th>Objective</th>
<th>$|\varepsilon_x^{(j)}|_\infty$</th>
<th>$|\varepsilon_q^{(j)}|_\infty$</th>
<th>Solve</th>
<th>$\varepsilon_x$</th>
<th>$\varepsilon_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_0$</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E$^{-2}$</td>
<td>2.443E$^{-2}$</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E$^{-4}$</td>
<td>1.444E$^{-3}$</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
</tr>
<tr>
<td>$\pi_{ML}$</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
</tr>
<tr>
<td>$\pi_F$</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td>9.7805377</td>
<td>3.996E$^{-5}$</td>
<td>–</td>
<td>4840.185</td>
<td>773.192</td>
<td>–</td>
</tr>
</tbody>
</table>

### Main remarks

- $\pi_{ML}$ has only $32.4\%$ of the nodes of $\pi_F$
- Local errors of the same order of magnitude
- Local errors of the high order of magnitude
### Multi–Level Adaptive Time–Mesh Refinement: Error analysis

| \( \pi_j \) | \( N_j \) | \( \Delta t_j \) | \( l_j \) | Objective | \( \| \varepsilon_x^{(j)} \|_{\infty} \) | \( \| \varepsilon_q^{(j)} \|_{\infty} \) | Solve CPU time (s) |
|---|---|---|---|---|---|---|---|---|
| \( \pi_0 \) | 101 | 1/100 | 25 | \( 9.7700173 \) | \( 1.029E^{-2} \) | \( 2.443E^{-2} \) | 1.859 | 0.638 | 0.004 |
| \( \pi_1 \) | 869 | 1/25600 | 34 | \( 9.7805247 \) | \( 4.096E^{-4} \) | \( 1.444E^{-3} \) | 8.895 | 6.345 | 0.007 |
| \( \pi_2 \) | 8285 | 1/25600 | 50 | \( 9.7805398 \) | \( 4.714E^{-5} \) | \( 1.684E^{-4} \) | 186.870 | 102.973 | 0.018 |
| \( \pi_{ML} \) | 8285 | 1/25600 | 109 | \( 9.7805398 \) | \( 4.714E^{-5} \) | \( 1.684E^{-4} \) | 197.624 | 109.956 | 0.029 |
| \( \pi_F \) | 25601 | 1/25600 | 406 | \( 9.7805377 \) | \( 3.996E^{-5} \) | \( 1.684E^{-4} \) | 4840.185 | 773.192 | – |

**Main remarks**

- \( \pi_{ML} \) has only 32.4% of the nodes of \( \pi_F \)
- Local errors of the same order of magnitude
- Solving OCP using \( \pi_{ML} \) is 38.4× faster than using \( \pi_F \)
- Local errors of the high order of magnitude
Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>$\pi_j$</th>
<th>$N_j$</th>
<th>$\Delta t_j$</th>
<th>$l_j$</th>
<th>Objective</th>
<th>$|\varepsilon_x^{(j)}|_{\infty}$</th>
<th>$|\varepsilon_q^{(j)}|_{\infty}$</th>
<th>CPU time (s)</th>
<th>$\varepsilon_x$</th>
<th>$\varepsilon_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_0$</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td></td>
<td>1.029E$^{-2}$</td>
<td>2.443E$^{-2}$</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td></td>
<td>4.096E$^{-4}$</td>
<td>1.444E$^{-3}$</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td></td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
</tr>
<tr>
<td>$\pi_{ML}$</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td></td>
<td>4.714E$^{-5}$</td>
<td>1.684E$^{-4}$</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
</tr>
<tr>
<td>$\pi_F$</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td></td>
<td>3.996E$^{-5}$</td>
<td>–</td>
<td>4840.185</td>
<td>773.192</td>
<td>–</td>
</tr>
<tr>
<td>$\pi_S$</td>
<td>8285</td>
<td>1/8284</td>
<td>80</td>
<td></td>
<td>1.236E$^{-4}$</td>
<td>–</td>
<td>333.428</td>
<td>113.912</td>
<td>–</td>
</tr>
</tbody>
</table>

Main remarks

- $\pi_{ML}$ has only 32.4% of the nodes of $\pi_F$
- Local errors of the same order of magnitude
- Solving OCP using $\pi_{ML}$ is 38.4× faster than using $\pi_F$
- Local errors of the high order of magnitude
## Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>( \pi_j )</th>
<th>( N_j )</th>
<th>( \Delta t_j )</th>
<th>( l_j )</th>
<th>Objective</th>
<th>( | \varepsilon_x^{(j)} |_\infty )</th>
<th>( | \varepsilon_q^{(j)} |_\infty )</th>
<th>Solve</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_0 )</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E−2</td>
<td>2.443E−2</td>
<td>1.859</td>
<td>0.638</td>
</tr>
<tr>
<td>( \pi_1 )</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E−4</td>
<td>1.444E−3</td>
<td>8.895</td>
<td>6.345</td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E−5</td>
<td>1.684E−4</td>
<td>186.870</td>
<td>102.973</td>
</tr>
<tr>
<td>( \pi_{ML} )</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E−5</td>
<td>1.684E−4</td>
<td>197.624</td>
<td>109.956</td>
</tr>
<tr>
<td>( \pi_F )</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td>9.7805377</td>
<td>3.996E−5</td>
<td>–</td>
<td>4840.185</td>
<td>773.192</td>
</tr>
<tr>
<td>( \pi_S )</td>
<td>8285</td>
<td>1/8284</td>
<td>80</td>
<td>9.7805577</td>
<td>1.236E−4</td>
<td>–</td>
<td>333.428</td>
<td>113.912</td>
</tr>
</tbody>
</table>

**Main remarks**

- \( \pi_{ML} \) has only 32.4% of the nodes of \( \pi_F \)
- **Local errors of the same order of magnitude**
- Solving OCP using \( \pi_{ML} \) is **38.4x faster** than using \( \pi_F \)
- \( \pi_{ML} \) has the same number of nodes of \( \pi_S \)
- **Local errors of the high order of magnitude**
### Multi–Level Adaptive Time–Mesh Refinement: Error analysis

| \( \pi_j \) | \( N_j \) | \( \Delta t_j \) | \( l_j \) | Objective | \( ||\varepsilon_x^{(j)}||_\infty \) | \( ||\varepsilon_q^{(j)}||_\infty \) | Solve | \( \varepsilon_x \) | \( \varepsilon_q \) | CPU time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| \( \pi_0 \) | 101 | 1/100 | 25 | 9.7700173 | 1.029E\(-2\) | 2.443E\(-2\) | 1.859 | 0.638 | 0.004 |
| \( \pi_1 \) | 869 | 1/25600 | 34 | 9.7805247 | 4.096E\(-4\) | 1.444E\(-3\) | 8.895 | 6.345 | 0.007 |
| \( \pi_2 \) | 8285 | 1/25600 | 50 | 9.7805398 | 4.714E\(-5\) | 1.684E\(-4\) | 186.870 | 102.973 | 0.018 |
| \( \pi_{ML} \) | 8285 | 1/25600 | 109 | 9.7805398 | 4.714E\(-5\) | 1.684E\(-4\) | 197.624 | 109.956 | 0.029 |
| \( \pi_F \) | 25601 | 1/25600 | 406 | 9.7805377 | 3.996E\(-5\) | – | 4840.185 | 773.192 | – |
| \( \pi_S \) | 8285 | 1/8284 | 80 | 9.7805577 | 1.236E\(-4\) | – | 333.428 | 113.912 | – |

**Main remarks**

- \( \pi_{ML} \) has only 32.4% of the nodes of \( \pi_F \)
- Local errors of the same order of magnitude
- Solving OCP using \( \pi_{ML} \) is 38.4\times faster than using \( \pi_F \)
- \( \pi_{ML} \) has the same number of nodes of \( \pi_S \)
- Local errors of the high order of magnitude
## Multi–Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>( \pi_j )</th>
<th>( N_j )</th>
<th>( \Delta t_j )</th>
<th>( I_j )</th>
<th>Objective</th>
<th>( |\varepsilon_x^{(j)}|_\infty )</th>
<th>( |\varepsilon_q^{(j)}|_\infty )</th>
<th>Solve</th>
<th>CPU time (s)</th>
<th>( \varepsilon_x )</th>
<th>( \varepsilon_q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_0 )</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E−2</td>
<td>2.443E−2</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>( \pi_1 )</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E−4</td>
<td>1.444E−3</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
<td></td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E−5</td>
<td>1.684E−4</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
<td></td>
</tr>
<tr>
<td>( \pi_{ML} )</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E−5</td>
<td>1.684E−4</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
<td></td>
</tr>
</tbody>
</table>

Main remarks

- \( \pi_{ML} \) has only 32.4% of the nodes of \( \pi_F \)
- Local errors of the same order of magnitude
- Solving OCP using \( \pi_{ML} \) is 38.4\times faster than using \( \pi_F \)
- \( \pi_{ML} \) has the same number of nodes of \( \pi_S \)
- Local errors of the high order of magnitude
- Solving OCP using \( \pi_{ML} \) is 1.7\times faster than using \( \pi_S \)
## Multi-Level Adaptive Time–Mesh Refinement: Error analysis

<table>
<thead>
<tr>
<th>( \pi_j )</th>
<th>( N_j )</th>
<th>( \Delta t_j )</th>
<th>( l_j )</th>
<th>Objective</th>
<th>( | \varepsilon_x^{(j)} |_\infty )</th>
<th>( | \varepsilon_q^{(j)} |_\infty )</th>
<th>Solve ( \varepsilon_x )</th>
<th>( \varepsilon_q )</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_0 )</td>
<td>101</td>
<td>1/100</td>
<td>25</td>
<td>9.7700173</td>
<td>1.029E(^{-2})</td>
<td>2.443E(^{-2})</td>
<td>1.859</td>
<td>0.638</td>
<td>0.004</td>
</tr>
<tr>
<td>( \pi_1 )</td>
<td>869</td>
<td>1/25600</td>
<td>34</td>
<td>9.7805247</td>
<td>4.096E(^{-4})</td>
<td>1.444E(^{-3})</td>
<td>8.895</td>
<td>6.345</td>
<td>0.007</td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>8285</td>
<td>1/25600</td>
<td>50</td>
<td>9.7805398</td>
<td>4.714E(^{-5})</td>
<td>1.684E(^{-4})</td>
<td>186.870</td>
<td>102.973</td>
<td>0.018</td>
</tr>
<tr>
<td>( \pi_{ML} )</td>
<td>8285</td>
<td>1/25600</td>
<td>109</td>
<td>9.7805398</td>
<td>4.714E(^{-5})</td>
<td>1.684E(^{-4})</td>
<td>197.624</td>
<td>109.956</td>
<td>0.029</td>
</tr>
<tr>
<td>( \pi_F )</td>
<td>25601</td>
<td>1/25600</td>
<td>406</td>
<td>9.7805377</td>
<td>3.996E(^{-5})</td>
<td>–</td>
<td>4840.185</td>
<td>773.192</td>
<td>–</td>
</tr>
<tr>
<td>( \pi_S )</td>
<td>8285</td>
<td>1/8284</td>
<td>80</td>
<td>9.7805577</td>
<td>1.236E(^{-4})</td>
<td>–</td>
<td>333.428</td>
<td>113.912</td>
<td>–</td>
</tr>
</tbody>
</table>

### Main remarks

- \( \pi_{ML} \) has only 32.4% of the nodes of \( \pi_F \)
- Local errors of the same order of magnitude
- Solving OCP using \( \pi_{ML} \) is 38.4\( \times \) faster than using \( \pi_F \)
- \( \pi_{ML} \) has the same number of nodes of \( \pi_S \)
- Local errors of the high order of magnitude
- Solving OCP using \( \pi_{ML} \) is 1.7\( \times \) faster than using \( \pi_S \)
Final Remarks

Main advantages of refinement strategies

► no need to define \textit{a priori} the most appropriately mesh
► local mesh resolution → nodes only where they are required
► initial coarse mesh → first solution obtained after few iterations
► warm–start after refinement → faster re–solve
► less nodes in the overall procedure → significant savings in memory and computational cost

Main advantages of our refinement algorithm

► $\pi_{ML}$: multi–level refinement → faster convergence
► $\pi_{ML}$: local error on $q$ → more accurate and faster error estimate
► $\pi_{ML}$: solution obtained very quickly (if the procedure ends in an early stage)
Final Remarks

Main advantages of refinement strategies

- no need to define \textit{a priori} the most appropriately mesh
- local mesh resolution → nodes only where they are required
- initial coarse mesh → first solution obtained after few iterations
- warm–start after refinement → faster re–solve
- less nodes in the overall procedure → significant savings in memory and computational cost

Main advantages of our refinement algorithm

- $\pi_{ML}$: multi–level refinement → faster convergence
- $\pi_{ML}$: local error on $q$ → more accurate and faster error estimate
- $\pi_{ML}$: solution obtained very quickly (if the procedure ends in an early stage)
Efficient solution of Continuous-Time optimal control problems

Mesh refinement in MPC

Numerical Solution of OCP

Mesh Refinement for Optimal Control

Mesh refinement in MPC
Algorithm Extension for MPC

▶ Question
  ▶ Why should we use a fine mesh to compute the solution in \([t_k, t_k + T]\) if we only implement the control in \([t_k, t_k + \delta]\)?

▶ Idea
  ▶ To use a mesh that is fine in the left–end and coarse in the right–end of \([t_k, t_k + T]\).

▶ Motivation
  ▶ Hourly planning
  ▶ Daily planning
  ▶ Monthly planning
Question

Why should we use a fine mesh to compute the solution in \([t_k, t_k + T]\) if we only implement the control in \([t_k, t_k + \delta]\)?

Idea

To use a mesh that is fine in the left–end and coarse in the right–end of \([t_k, t_k + T]\).

Motivation

- Hourly planning
- Daily planning
- Monthly planning
Algorithm Extension for MPC

▶ Question
  ▶ Why should we use a fine mesh to compute the solution in \([t_k, t_k + T]\) if we only implement the control in \([t_k, t_k + \delta]\)?

▶ Idea
  ▶ To use a mesh that is fine in the left–end and coarse in the right–end of \([t_k, t_k + T]\).

▶ Motivation
  ▶ Hourly planning
  ▶ Daily planning
  ▶ Monthly planning
Algorithm Extension for MPC

- past feedback
- current time
- prediction horizon
- optimal control sequence

$\tilde{x}$
$past$
$\bar{x}$
$\star$
$u^*$
$past$ feedback
$u^*$ optimal control sequence

$t_0 \rightarrow t_k \rightarrow t_k + \delta \rightarrow T$
Algorithm Extension for MPC

Efficient solution of Continuous-Time optimal control problems

Mesh refinement in MPC
MPC with adaptive mesh: Error analysis

Table: Comparing MPC results for the problem $P$

<table>
<thead>
<tr>
<th>$\pi_j$</th>
<th>$N_j$</th>
<th>$\Delta t_j$</th>
<th>$I_j$</th>
<th>$|\varepsilon_x^{(j)}|_\infty$</th>
<th>CPU time (s)</th>
<th>Solver</th>
<th>$\varepsilon_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_{ML}$</td>
<td>$365 , 1/3200$</td>
<td>$304</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>$\pi_F$</td>
<td>$3201 , 1/3200$</td>
<td>$371</td>
<td>34</td>
<td>22</td>
<td>20</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>$\pi_C$</td>
<td>$201 , 1/200$</td>
<td>$233</td>
<td>81</td>
<td>13</td>
<td>11</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

- $\pi_{ML}$ has only $11.4\%$ of the nodes of $\pi_F$ and local error of the same order of magnitude.
- $\pi_{ML}$ takes less than $20\%$ of the time needed for $\pi_F$ to get a solution.
- $\pi_C$ (the initial coarse mesh with equidistant spacing) has lower accuracy, $1.261E^{-3}$, when compared against $4.169E^{-5}$.
- CPU time spent to compute solution using $\pi_{ML}$ is, as expected, $50\%$ higher. However it is a good trade–off gaining two orders of magnitude in the accuracy.
Further details:

Thank y’all!

faf@fe.up.pt,
http://www.fe.up.pt/~faf
Wisenbaker 333B