A computationally efficient model predictive control scheme for space debris rendezvous

A.K. Larsén

Y. Chen

M. Bruschetta R. Carli A. Cenedese

D. Varagnolo

L. Felicetti

Luleå University of Technology, Sweden
Eindhoven University of Technology, The Netherlands
University of Padova, Italy
Norwegian University of Science and Technology, Norway
Cranfield University, United Kingdom

Space Debris Removal - why

J.-C. Liou et al. / Acta Astronautica 66 (2010) 648–653 651

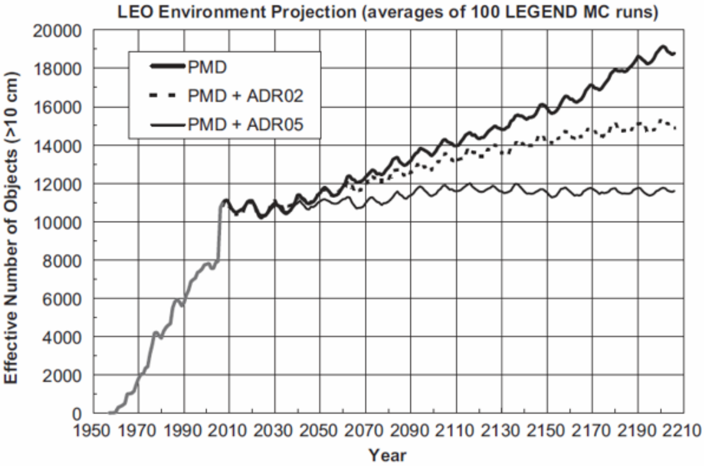
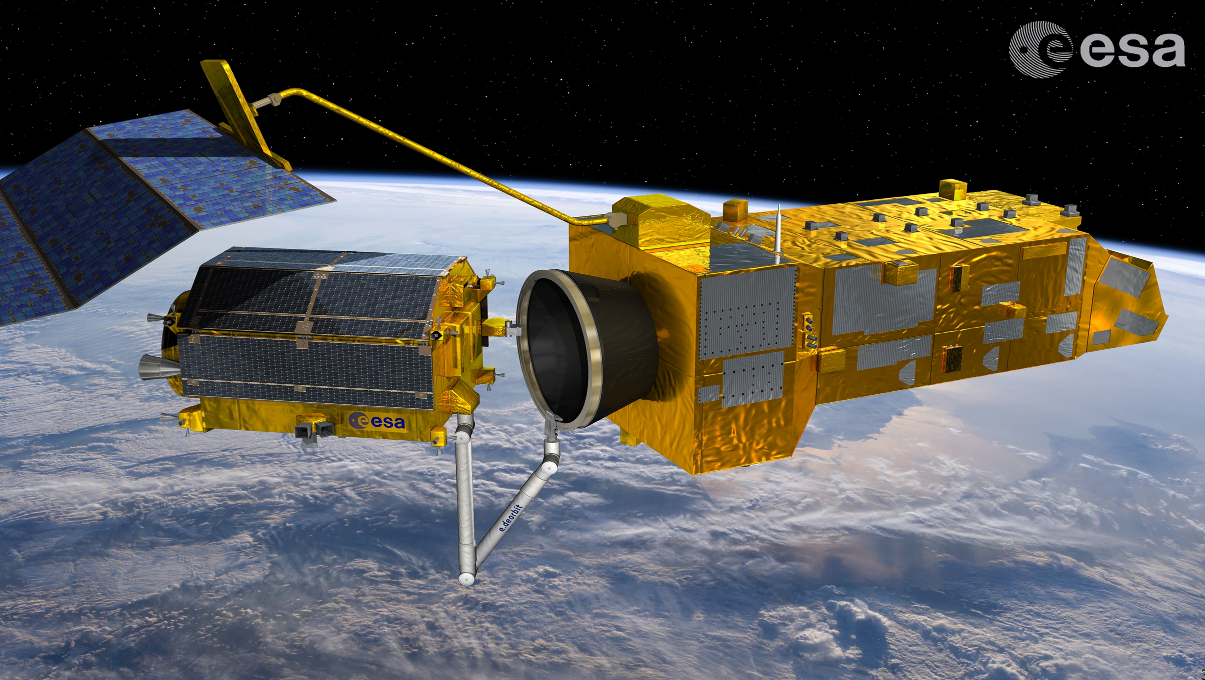
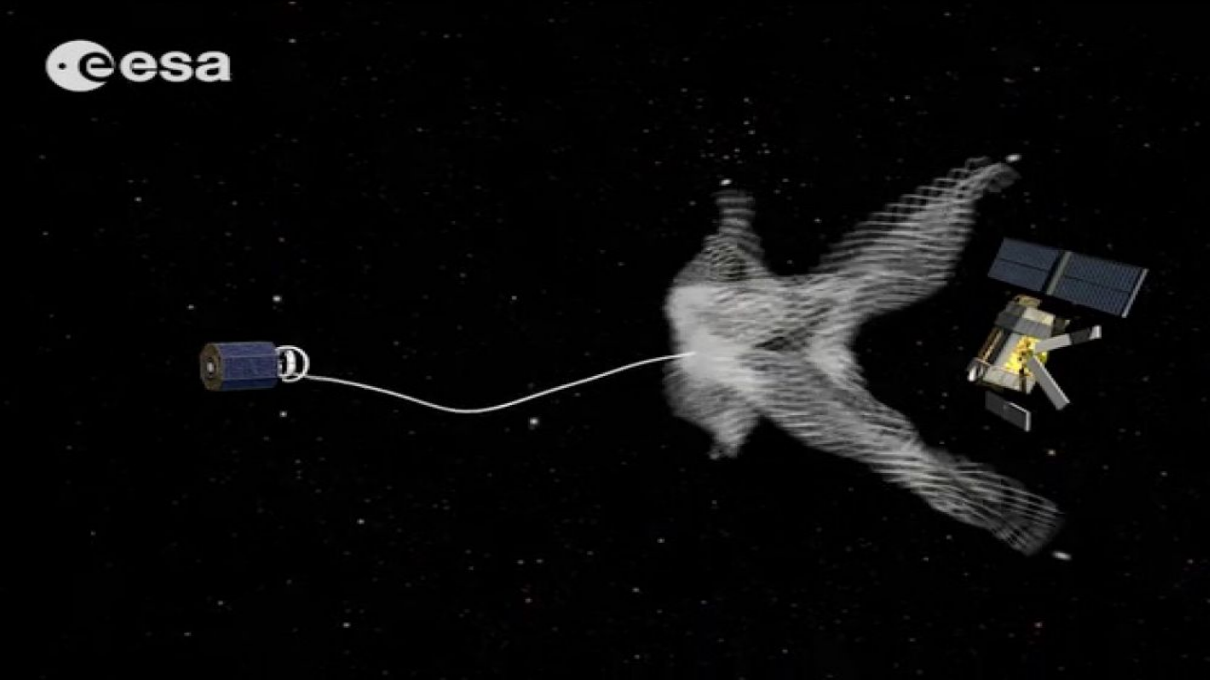


Fig. 2. Comparison of three different scenarios. From top to bottom: postmission disposal (PMD) only, PMD and ADR of two objects per year, and PMD and ADR of five objects per year, respectively.


esa
esa
e.deorbit


esa

overarching need: get spacecrafts close enough to non-collaborative debris

Key challenge for space rendezvous with debris

- better to use small satellites to reduce financial costs

  $\Longrightarrow$ limited actuation/thrust capabilities

  $\Longrightarrow$ long maneuvering times

implied need: calculate the on-board thrusters scheduling
taking into account the actuation limits

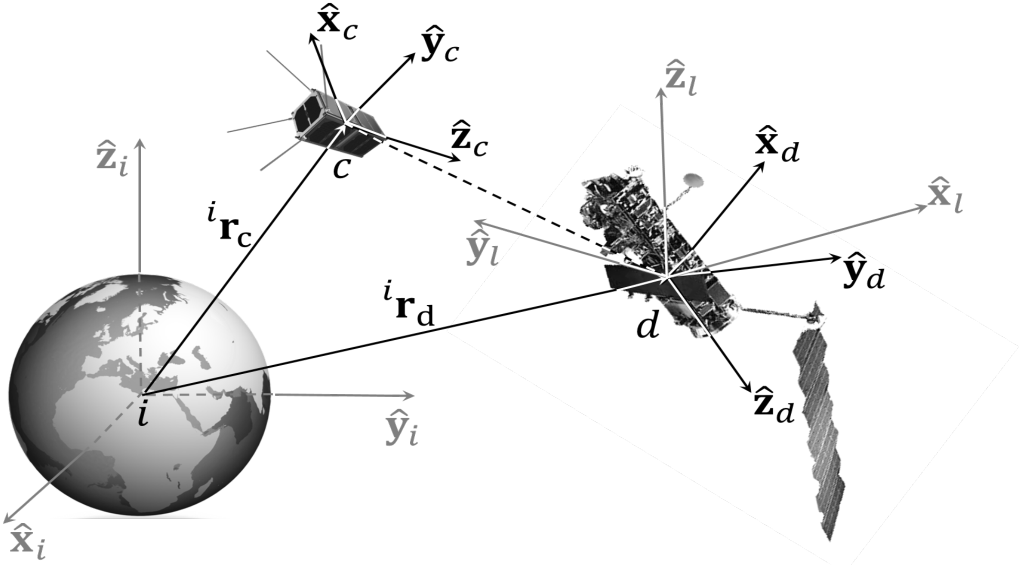
This study (in general)

cast the control problem as a *Model Predictive Control (MPC)* one,
but take into special account the *limited thrusting capabilities* of small satellites

This study (more specifically)

- formulate the MPC so that it:
  - accounts for non-linear orbital dynamics
  - emphasizes fuel consumption minimization
  - is numerically fast, so to achieve longer prediction horizons

Scenario



$\hat{\mathbf{x}}_c$
$\hat{\mathbf{y}}_c$
$\hat{\mathbf{z}}_c$
c
$\hat{\mathbf{z}}_l$
$\hat{\mathbf{x}}_d$
$\hat{\mathbf{x}}_l$
$\hat{\mathbf{y}}_l$
$\hat{\mathbf{y}}_d$
$\hat{\mathbf{z}}_i$
${}^i\mathbf{r}_c$
${}^i\mathbf{r}_d$
d
$\hat{\mathbf{z}}_d$
i
$\hat{\mathbf{y}}_i$
$\hat{\mathbf{x}}_i$

System Modeling: Orbital Dynamics

- Kinematics: ${}^{i}\dot{\boldsymbol{r}}_j = {}^{i}\boldsymbol{v}_j$
- Dynamics:

$${}^{i}\dot{\boldsymbol{v}}_j = {}^{i}\boldsymbol{a}_{g,j} + {}^{i}\boldsymbol{a}_{J2,j} + {}^{i}\boldsymbol{a}_{drag,j} + {}^{i}\boldsymbol{a}_{srp,j} + \frac{{}^{i}\boldsymbol{T}_j}{m_j}$$

where:

- $j = \{c, d\}$ (chaser or debris)
- ${}^{i}\boldsymbol{a}_{g,j} = -\mu\frac{\boldsymbol{r_j}}{{r_j}^3}$ with $\mu =$ Earth's gravitational parameter
- ${}^{i}\boldsymbol{a}_{J2,j}$: Earth's oblateness perturbation
- ${}^{i}\boldsymbol{a}_{drag,j}$: atmospheric drag perturbation
- ${}^{i}\boldsymbol{a}_{srp,j}$: solar radiation pressure perturbation
- m_j: mass
- ${}^{c}\boldsymbol{T}_c$: thrust (note: ${}^{i}\boldsymbol{T}_d = 0$)

Notation:

- ${}^{i}\boldsymbol{r}_j$ = orbital position
- ${}^{i}\boldsymbol{v}_j$ = orbital velocity
- $\boldsymbol{q}_j$ = attitude quaternion
- ${}^{j}\boldsymbol{\omega}_j$ = angular rate

System Modeling: Attitude Dynamics

- Kinematics: $\dot{\boldsymbol{q}}_j = \frac{1}{2} {}^j\boldsymbol{\omega}_j \otimes \boldsymbol{q}_j$
- Dynamics:

$$ {}^j\dot{\boldsymbol{\omega}}_j = \boldsymbol{I}_j^{-1}(-{}^j\boldsymbol{\omega}_j \times \boldsymbol{I}_j {}^j\boldsymbol{\omega}_j + {}^j\boldsymbol{\tau}_{gg,j} + {}^j\boldsymbol{\tau}_j) $$

where:

- $\otimes$: quaternion product
- $\boldsymbol{I}_j$: inertia matrix
- ${}^j\boldsymbol{\tau}_{gg,j}$: gravity gradient torque
- ${}^j\boldsymbol{\tau}_j$: control torque

Notation:

- ${}^i\boldsymbol{r}_j$ = orbital position
- ${}^i\boldsymbol{v}_j$ = orbital velocity
- $\boldsymbol{q}_j$ = attitude quaternion
- ${}^j\boldsymbol{\omega}_j$ = angular rate

note: no torques due to drag and solar radiation pressure

Differences between simulation vs. control models

	simulator	*controller*
atmospheric drag	✓	
solar radiation pressure	✓	
J_2 effects	✓	
imperfect knowledge about mass distributions	✓	
imperfect actuation	✓	
measurement noise	✓	
mass consumption during maneuvers		

State space representation of the chaser

state: $\boldsymbol{x} := \begin{bmatrix} {}^{i}\boldsymbol{r}_c^T & {}^{i}\boldsymbol{v}_c^T & \boldsymbol{q}_c^T & {}^{c}\boldsymbol{\omega}_c^T & {}^{c}\boldsymbol{T}_c^T & {}^{c}\boldsymbol{\tau}_c^T \end{bmatrix}^T$

inputs: $\boldsymbol{u} := \begin{bmatrix} {}^{c}\dot{\boldsymbol{T}}_c^T & {}^{c}\dot{\boldsymbol{\tau}}_c^T \end{bmatrix}^T$

in words:

state $=$ orbital position & velocity,
attitude, angular rate, applied thrusts and torques

inputs $=$ rate of variation of the applied thrusts and torques

Mission = make the chaser match its orbit with that of the debris

Mission = make the chaser match its orbit with that of the debris

$$\Longrightarrow \text{ relative dynamics: } \begin{cases} {}^{l}\boldsymbol{d}_c := {}^{l}\boldsymbol{R}_i\left({}^{i}\boldsymbol{r}_c - {}^{i}\boldsymbol{r}_d\right) & \textit{(position)} \\ {}^{l}\dot{\boldsymbol{d}}_c := {}^{l}\boldsymbol{R}_i\left({}^{i}\boldsymbol{v}_c - {}^{i}\boldsymbol{v}_d\right) + {}^{i}\boldsymbol{\omega}_l \times {}^{l}\boldsymbol{R}_i\left({}^{i}\boldsymbol{r}_c - {}^{i}\boldsymbol{r}_d\right) & \textit{(velocity)} \\ \boldsymbol{q}_{err} := \boldsymbol{q}_{c,ref} \otimes \boldsymbol{q}_c^* & \textit{(attitude error)} \end{cases}$$

with

${}^{c}\boldsymbol{R}_i =$ rotation matrix from the ECI to the LHLV reference frame

${}^{i}\boldsymbol{\omega}_l =$ angular velocity of the LHLV with respect to the ECI reference frame

Mission = make the chaser match its orbit with that of the debris

$$\implies \text{relative dynamics: } \begin{cases} {}^l\boldsymbol{d}_c := {}^l\boldsymbol{R}_i\big({}^i\boldsymbol{r}_c - {}^i\boldsymbol{r}_d\big) & \textit{(position)} \\ {}^l\dot{\boldsymbol{d}}_c := {}^l\boldsymbol{R}_i\big({}^i\boldsymbol{v}_c - {}^i\boldsymbol{v}_d\big) + {}^i\boldsymbol{\omega}_l \times {}^l\boldsymbol{R}_i\big({}^i\boldsymbol{r}_c - {}^i\boldsymbol{r}_d\big) & \textit{(velocity)} \\ \boldsymbol{q}_{err} := \boldsymbol{q}_{c,ref} \otimes \boldsymbol{q}_c^* & \textit{(attitude error)} \end{cases}$$

with

$^c\boldsymbol{R}_i$ = rotation matrix from the ECI to the LHLV reference frame

$^i\boldsymbol{\omega}_l$ = angular velocity of the LHLV with respect to the ECI reference frame

controller's goal = make $\boldsymbol{y} := \begin{bmatrix} {}^l\boldsymbol{d}_c & {}^l\dot{\boldsymbol{d}}_c & \boldsymbol{q}_{err} \end{bmatrix} \to \boldsymbol{0}$

Formulation as an optimal control problem

$$
\begin{aligned}
\min_{\widehat{\boldsymbol{u}}} \quad & \int_0^T \left(\|\widehat{\boldsymbol{y}}\|_Q^2 + \|\widehat{\boldsymbol{u}}\|_R^2 \right) d\tau \\
\text{s.t} \quad & \widehat{\boldsymbol{x}}_0 = \boldsymbol{x}(t) \\
& \widehat{\dot{\boldsymbol{x}}} = f\left(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{u}}\right) \\
& \widehat{\boldsymbol{y}} = h\left(\widehat{\boldsymbol{x}}, {}^i\widehat{\boldsymbol{r}}_d, {}^i\widehat{\boldsymbol{v}}_d, \boldsymbol{q}_{c,ref}\right) \\
& {}^c\widehat{\boldsymbol{T}}_c \in \left[\underline{{}^c\boldsymbol{T}_c} \quad \overline{{}^c\boldsymbol{T}_c}\right] \\
& {}^c\widehat{\boldsymbol{\tau}}_c \in \left[\underline{{}^c\boldsymbol{\tau}_c} \quad \overline{{}^c\boldsymbol{\tau}_c}\right] \\
& {}^c\widehat{\boldsymbol{\omega}}_c \in \left[\underline{{}^c\boldsymbol{\omega}_c} \quad \overline{{}^c\boldsymbol{\omega}_c}\right] \\
& {}^c\widehat{\dot{\boldsymbol{T}}}_c \in \left[\underline{{}^c\dot{\boldsymbol{T}}_c} \quad \overline{{}^c\dot{\boldsymbol{T}}_c}\right] \\
& {}^c\widehat{\dot{\boldsymbol{\tau}}}_c \in \left[\underline{{}^c\dot{\boldsymbol{\tau}}_c} \quad \overline{{}^c\dot{\boldsymbol{\tau}}_c}\right]
\end{aligned}
$$

$\rightarrow$ computational demands

...in discrete time

$$
\begin{aligned}
\min_{x_k, u_k} \; & \sum_{k=0}^{N-1} d\big(h(x_k, u_k) - h_{ref}^k\big)_{W_k} + d_N\big(h_N(x_N) - h_{ref}^N\big)_{W_N} \\
\text{s.t.} \; & 0 = x_0 - \hat{x}_0 \\
& 0 = x_{k+1} - \phi_k(x_k, u_k) \\
& \underline{x}_k \leq x_k \leq \overline{x}_k \\
& \underline{u}_k \leq u_k \leq \overline{u}_k \\
& \underline{r}_k \leq r_k(x_k, u_k) \leq \overline{r}_k
\end{aligned}
$$

with $\phi_k(x_k, u_k)$ a numerical integrator simulating

$$
0 = f\big(\dot{x}(t), x(t), u(t), t\big) \qquad x(0) = x_k
$$

Our strategy: use MATMPC

MATMPC

Matlab
(relatively slow)

C++ libraries
(faster but less straightforward)

Our strategy: use MATMPC

MATMPC

Matlab
(relatively slow)

C++ libraries
(faster but less straightforward)

MATMPC:

- based on MATLAB
- modular structure
- core modules written in MATLAB C API
- not a C/C++ library, but a combination of MATLAB scripts and MEX functions
- computationally fast (comparable to low level languages)

Supported algorithms

Hessian Approximation	Generalized Gauss-Newton		
Integrator	Explicit Runge Kutta 4 (`CasADi` code generation)	Explicit Runge Kutta 4	Implicit Runge-Kutta (Gauss-Legendre)
Condensing	non	full	partial
QP solver	`qpOASES` `OSQP`	`MATLAB quadprog` `HPIPM`	`Ipopt`
Globalization	ℓ_1 merit function line search		Real-Time Iteration
Special features	CMoN-SQP	input move blocking	non-uniform grid

implication: can consider longer prediction horizons

$\Longrightarrow$ better divide the mission in distance-depending stages

Indeed:

when too far from the debris. . .

- . . . info about the target's motion are of limited quality

  $\Longrightarrow$ following exactly the reference trajectory may be fuel-inefficient

  & better to have longer sampling periods

Indeed:

when too far from the debris...

- ...info about the target's motion are of limited quality

  $\implies$ following exactly the reference trajectory may be fuel-inefficient
  & better to have longer sampling periods

when close to the target...

- ...need to prepare for the docking

  $\implies$ emphasize precision instead of fuel-consumption
  & use shorter sampling periods

Indeed:

when too far from the debris. . .

- . . . info about the target's motion are of limited quality

  $\Longrightarrow$ following exactly the reference trajectory may be fuel-inefficient
  & better to have longer sampling periods

when close to the target. . .

- . . . need to prepare for the docking

  $\Longrightarrow$ emphasize precision instead of fuel-consumption
  & use shorter sampling periods

> chosen approach: use two MPCs with the same dynamics
> and different prediction horizons / cost functions

What do we want to understand?

How do. . .

- the length of the prediction horizon
- the definition of *"close"* vs *"far"*

. . . affect the mission efficiency? *(fuel and time)*

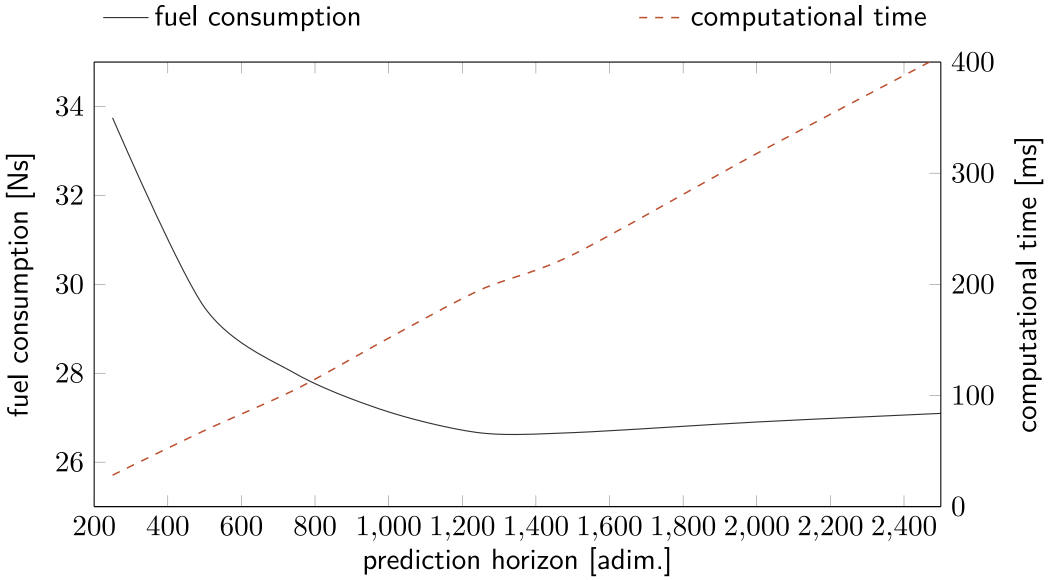
Simulation Parameters - scenario

- Space debris:
  - in circular orbit
  - altitude: $300km$
  - inclination: $30\,\deg$
- Chaser:
  - 3U CubeSat
  - same orbital parameters as space debris
  - mass: $4kg$
  - inertia matrix:

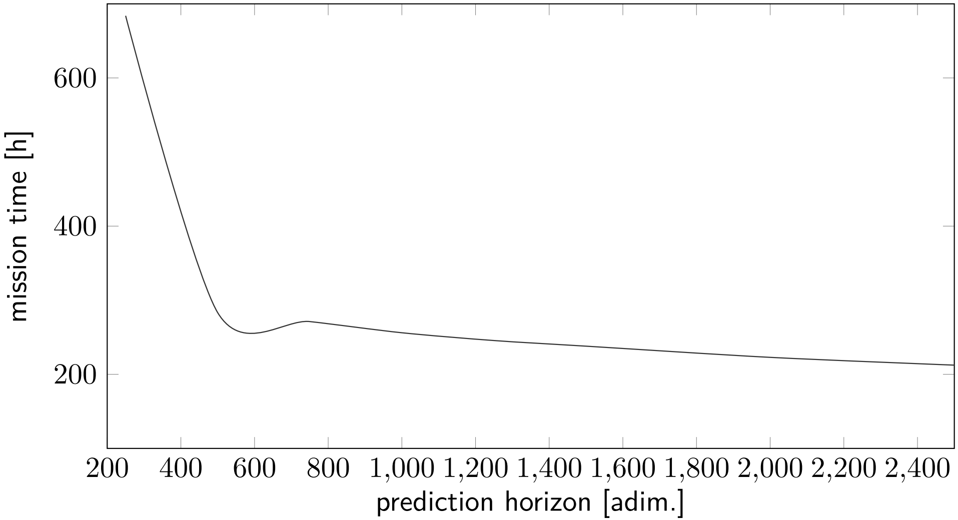
$$I_c = \begin{bmatrix} 0.0333 & 0.0000 & 0.0000 \\ 0.0000 & 0.0067 & 0.0000 \\ 0.0000 & 0.0000 & 0.0333 \end{bmatrix} kg\, m^2$$

  - *true anomaly offset:* $\Delta\nu = -10\,\deg \implies$ *initial distance from the debris:* $1164km$

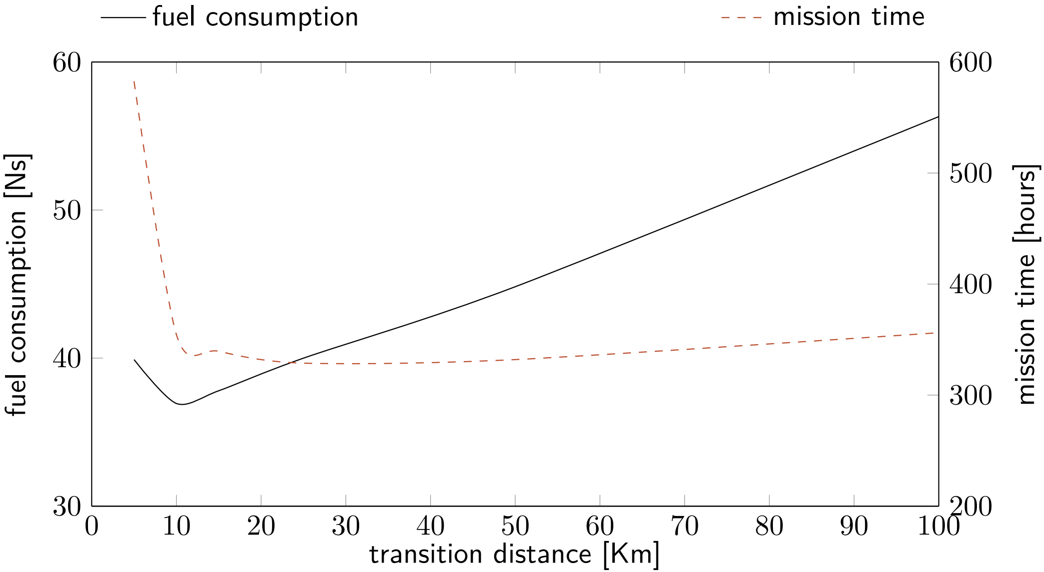
Prediction horizon length vs mission efficiency (1)



Prediction horizon length vs mission efficiency (2)



Transition distance vs mission efficiency



Conclusions

- if having low actuation capabilities, then better to increase the prediction-horizons
- long prediction-horizons require computational efficiency
- when considering long missions we should use time-varying controllers
  ($\implies$ *worth to be explored further*)

A computationally efficient model predictive control scheme for space debris rendezvous

A.K. Larsén
Y. Chen
M. Bruschetta R. Carli A. Cenedese
D. Varagnolo
L. Felicetti

Luleå University of Technology, Sweden
Eindhoven University of Technology, The Netherlands
University of Padova, Italy
Norwegian University of Science and Technology, Norway
Cranfield University, United Kingdom