Control System for Free-Floating Space Manipulator with NMPC: Implementations and Extensions

Daniel Vedova
Robotics Institute
dkv@cs.cmu.edu

Thomas Moriarty
CMU Mechanical Engineering
tmoriart@andrew.cmu.edu

I. Introduction

The work by Tomasz Rybus et. al. in [8] explores maneuvers of a floating base satellite with an open chain manipulator to interact with a separate moving body. The study performs a trajectory optimization on a representative planar satellite model, and implements the trajectory using non-linear Model Predictive Control. The present work builds a re-creation of the methods and results presented in the previous work, and explores alternative MPC control applied to the same motion scenarios.

To implement and explore the proposed paper of study [8], a representative system model was created using the SPART toolbox. The model allows for state dynamics description of a floating base manipulator system. Trajectory optimizations were performed on the models, including a representation of the rendezvous action presented in the work. Differing instances of model predictive control were then applied to the generated trajectory, and compared for different cost weightings and model discrepancies.

Extensions of the methods explored in the paper were also performed on a simple pendulum system exhibiting non-linear system dynamics. A comparison of adaptive and non-linear Model Predictive Control techniques was performed using the pendulum model.

The report is structured in the following sections. Section II describes the target application of the work. Section III provides supporting description and theory for the tools and methods used in the report,with section III-A detailing the system modelling and III-B formulating the MPC methods explored. Section IV discusses the trajectory planning and optimization of the model from III-A. Section V presents the results of trajectory tracking using different control methods. A initial comparison of model predictive control methods on simple pendulums is shown in section V-A, and adaptive MPC and non-linear MPC are compared on the satellite model in V-B. Concluding remarks are presented in VI.

II. Motivation

There is significant commercial value in finding new ways to extend the life or repair aging satellites in orbit. One such method to extend the lives of satellites in orbit is to launch a spacecraft with a mounted manipulator, which can dock with other satellites and perform servicing and repair tasks. Studying control methods for floating base manipulators is a key step in realizing repair/rehabilitation goals for aging satellites. This technology dates back to the 1980s when the first space manipulator mission was launched using the Shuttle Remote Manipulator System (SRMS), which is also known as the Canadarm [10]. Another notable technology demonstration is the Japanese ETS-VII mission [6], which demonstrated a real life docking/rendezvous capability for space robots in 1997. The planning and control of floating manipulator actions is complicated by conservation of momentum, resulting in inertial dependencies in the motion. While an actuated satellite base can account for the required reaction forces to simulate a fixed base scenario, this action requires fuel; a costly trade for base control. Therefore planning and control of the manipulator with a passive base offers operational benefits in the form of reduced energy consumption.

III. Background

A. *system models*

1) Floating base dynamics: Here we discuss the dynamics of a floating base manipulator. Because the base of the robot is free floating, we must include the position and orientation of the base in the state description, as can be seen in equation (3). We modeled our system without an actuated base, and so our system is trivially underactuated. The underactuated nature of our system makes computing the dynamics of the system more complicated, because our system must obey conservation of angular and linear momentum laws. Our system is governed by the following equations of motion:

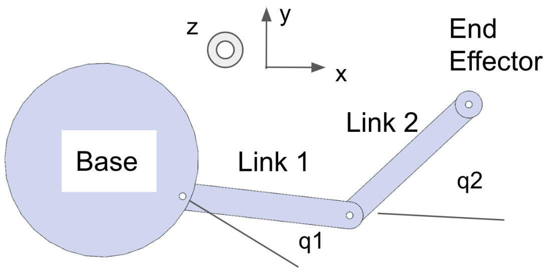


Fig. 1: Planar model

$$\begin{bmatrix} H_b & H_c \\ H_c^T & H_m \end{bmatrix} \ddot{q} + \begin{bmatrix} c_b \\ c_m \end{bmatrix} = \tau \tag{1}$$

Where H_b is the inertia matrix of the base, H_c is the coupling inertia matrix between the base and the manipulator, and H_m is the manipulator inertia matrix. The coupling inertia matrix helps describe how the motion of the base affects the motion of the arm, and how the motion of the arm affects motion of the base.

2) SPART Toolbox: The Space Robotic Toolbox (SPART) [9] is a package developed for kinematic and dynamic analysis of open-chain manipulators. The MATLAB-based toolbox accepts a Unified Robot Description Format (URDF) representation of a manipulator system, and provides analysis tools for the provided model, allowing for both fixed-base and floating-base systems. Dynamics are solved using a recursive Newton-Euler algorithm (RNEA) using a Natural Orthogonal Compliment method.

Figure 2 depicts the formulation of state dynamics used in the study. Forward dynamics relate the current state and control inputs to the requisite state rate of change. The generalized state vector is analyzed using forward kinematics and differential kinematics, which creates a representation of configuration specific inertial relations. The applied control inputs are then combined to complete a forward dynamic calculation using RNEA to return Joint and base accelerations. The forward dynamics are used for both trajectory optimization and simulation.

3) Planar space robot: In order to analyze the relevant operations presented in [8], a dynamic model was created using a URDF and the SPART toolbox. Figure 1 and Table I describe the system used for planar analysis throughout the present study. The model consists of three bodies: a base and two links in an open-chain configuration. The two consecutive joints connecting the three bodies allow for rotational motion about parallel axes, constraining the active motions to a plane. Each joint is active, providing a control vector of $u = [T_{j1}, T_{j2}]^T$. The base is unconstrained to the inertial frame, which

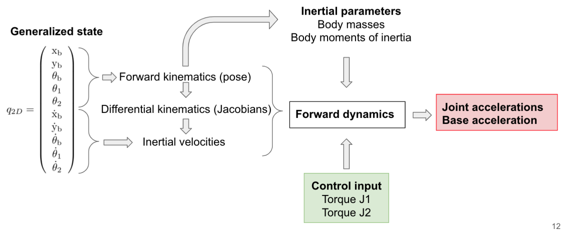

Generalized state
$q_{2D} = (x_b, y_b, \theta_b, \theta_1, \theta_2, \dot{x}_b, \dot{y}_b, \dot{\theta}_b, \dot{\theta}_1, \dot{\theta}_2)$
Forward kinematics (pose)
Differential kinematics (Jacobians)
Inertial velocities
Inertial parameters
Body masses
Body moments of inertia
Forward dynamics
Joint accelerations
Base acceleration
Control input
Torque J1
Torque J2
12

Fig. 2: Dynamics flowchart

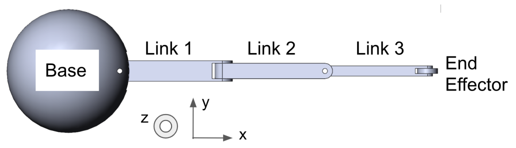


Fig. 3: Spatial model

introduces three additional coordinates for state description. This condition leads to an underactuated system, with minimum five generalized state coordinates, and two independent actuated inputs.

Parameter	value	unit
Base mass	12.9	kg
Base moment of inertia	0.208	kgm^2
Base center to joint 1	0.327	m
Link 1 mass	4.5	kg
Link 1 moment of inertia	0.32	kgm^2
Link 1 length	0.62	m
Link 2 mass	1.5	kg
Link 2 moment of inertia	0.049	kgm^2
Link 2 length	0.6	m

TABLE I: Parameters for the major components of the planar model satellite

$$q_{Planar} \in \mathbb{R}^{10} : \tag{2}$$

$$q_{Planar} = [\mathrm{x_b}, \mathrm{y_b}, \theta_\mathrm{b}, \theta_1, \theta_2, \dot{\mathrm{x}}_\mathrm{b}, \dot{\mathrm{y}}_\mathrm{b}, \dot{\theta}_\mathrm{b}, \dot{\theta}_1, \dot{\theta}_2] \tag{3}$$

4) URDF for spatial robot: Figure 3 depicts an extended model for further study, including a fourth body as a third link for movement out of plane. The orthogonal joint allows for spatial operation. This report does not provide an in depth study of spatial operation, however the methods presented are valid for the extension, and is a target for future work.

5) Augmented state: Tasks required of the system generally relate to the end effector state. Since the generalized coordinates for describing the system dynamics do not directly provide end effector information, an augmented state with explicit end effector position and velocity in the inertial coordinate frame was created. The augmented state allows access to end effector state for trajectory optimization and control. The transformation from the minimal state to the end effector position and velocity is

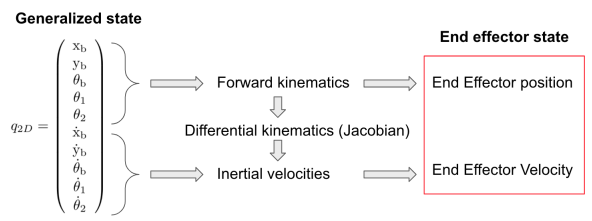

Generalized state
$q_{2D} =$
x_b y_b θ_b θ_1 θ_2 $\dot{x}_b$ $\dot{y}_b$ $\dot{\theta}_b$ $\dot{\theta}_1$ $\dot{\theta}_2$
Forward kinematics
Differential kinematics (Jacobian)
Inertial velocities
End effector state
End Effector position
End Effector Velocity

Fig. 4: Dynamics flowchart

shown in figure 4. The augmented state vector is shown in equation 4.

$q_{PlanarAug} \in \mathbb{R}^{14}$:

$$q_{PlanarAug} = \begin{pmatrix} \mathrm{x_b} \\ \mathrm{y_b} \\ \theta_\mathrm{b} \\ \theta_1 \\ \theta_2 \\ \dot{\mathrm{x}}_\mathrm{b} \\ \dot{\mathrm{y}}_\mathrm{b} \\ \dot{\theta}_\mathrm{b} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \mathrm{EE_{posX}} \\ \mathrm{EE_{posY}} \\ \mathrm{EE_{velX}} \\ \mathrm{EE_{velY}} \end{pmatrix} \tag{4}$$

B. Model Predictive Control

Model Predictive Control (MPC) is a control method that produces the optimal control action at each time step during operation through solving a local optimization problem[3] at that time step. Figure 5 depicts a temporal representation of the MPC process. The optimization is performed over a prediction horizon into the future, which shifts forward during operation as a "receding horizon." Different methods for model predictive control address trade-offs between optimization accuracy, horizon distance, cost definition, constraint definitions, and their effect on computational load. Model predictive controllers are model-based controllers, relying on the accuracy of the model relative to the representative system for efficacy. MPC methods differ from other optimal control techniques such as Linear Quadratic Regulation due to the run-time iterative process of optimization. MATLAB provides a set of tools for implementing model predictive controllers.[2]

1) Linear MPC: Linear MPC performs the predictive control process through Posing a quadratic cost subject to a linear state dynamic relationship and linear constraints on state and control. The optimization problem is solved using quadratic program methods.

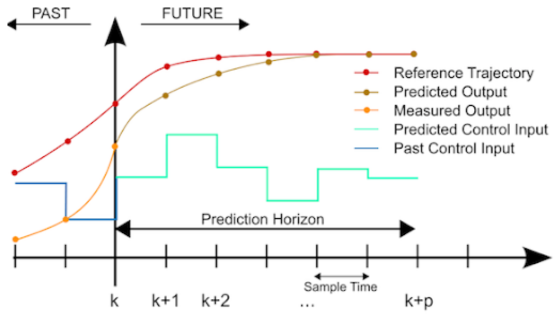


Fig. 5: MPC graphical example

The following equations describe the linear MPC method. Cost associated with the optimization is defined in the form of equation 5 and 6, where the cost terms J_y, J_u, $J_{\Delta u}$, and J_ϵ denote costs for output reference error, control effort, control effort rate of change, and soft constraint violations, respectively. The total cost is represented by equation 5.

$$J = J_y + J_u + J_{\Delta u} + J_\epsilon \tag{5}$$

The quadratic form of the cost function shown in 6, where k represents the current time step, and p the prediction horizon for optimization.

$$J = \sum_{i=k}^{k+p} W(x_{k+i} - x_{d,k+i})^2 \tag{6}$$

A system with linear dynamics can be represented in the classical state-space form 7.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{7}$$

The quadratic form of the cost function is convex; a form well suited to optimization. Given a positive definite set of weightings, the quadratic cost and linear dynamics, the optimization can be formed into a quadratic program III-B1 with linear constraints, and can be solved using efficient methods. H and f are the Hessian and gradient of the objective function.

$$\begin{aligned} &\min_x \left(\frac{1}{2}u^T H u + f^T u\right) \\ &\text{subject to} : Ax = b \end{aligned}$$

The unconstrained version of the problem can be solved analytically, while the constrained problem can be solved using numerical methods. Common methods for solving the constrained QP are Interior point solvers and Active-set solvers. The MATLAB MPC toolbox uses an Active set solver as the default option for optimization.

2) Adaptive MPC: Linear MPC methods require an adequately Linear time-invariant system for viable operation. For the system described in III-A, the state dynamic equations exhibit configuration dependent relations, forming a significantly non-linear system over the space of configurations. Non-linear systems pose more complicated dynamics that in general cannot be analyzed directly with linear methods. Adaptive MPC addresses control of non-linear systems using linear MPC methods through applying linearizations of the dynamics at each time step. While continual linearizations add computational load to the system, the simplification of the optimization to a linear system significantly reduces the total decision load per time step. A linearization is only valid around an equilibrium condition of the system at which the approximation is applied. Thus for good performance, the relationship between the reference target and system state should be within reasonable proximity (trust region). Given a marginal state deviation over the time horizon to the reference value, a linear approximation of the non-linear dynamics can provide adequate dynamic accuracy to implement linear MPC methods on the linearized system.

Equations 8-10 describe the linearization process[7]. Both the nominal state and control describe the linearization location, and the partial differentiation of the dynamics with respect to the state and control provides the first order description of the small deviation away from the local point.

$$\dot{\bar{x}} = f(\bar{x}, \bar{u}) \tag{8}$$

$$x(t) = \bar{x}(t) + \delta_x(t) \tag{9}$$

$$\dot{\bar{x}}(t) + \dot{\delta_x}(t) \approx f(\bar{x}, \bar{u}) + \frac{\partial f}{\partial x}|_{\bar{x},\bar{u}}\delta_x(t) + \frac{\partial f}{\partial u}|_{\bar{x},\bar{u}}\delta_u(t) \tag{10}$$

The MPC process can now perform optimization using the linear approximation as a replacement for the full non-linear model. In order to obtain the linearized system dynamics without an explicit analytical model, numerical differentiation can be performed using the method of finite differences. The MATLAB toolbox Adaptive Robust Numerical Differentiation[1] provides functions for generating the linearized dynamic equations with respect to state and control at an operating point, and is used for linearization in this work.

3) Non-linear MPC: When the posed control problem exhibits substantially non-linear dynamics, non-linear or time varying constraints, and alternative cost functions that don't assume a globally convex form, the model predictive control process requires more sophisticated tools to perform the optimization. Generally an iterative optimization process is used to minimize the cost function, such as sequential quadratic programming (SQP), which serves as the default solving method for MATLAB NMPC toolbox. It is important to note the iterative optimization described here is iterated within the current time step to find the optimal control action, leading to substantial increase in computational demands for the controller.

Sequential quadratic optimization method [4]

1) Approximate optimization with a quadratic form
2) Solve the Quadratic Program sub-problem
3) iterate process at the new location
4) Terminate process when minimization is found

IV. Trajectory optimization

In order to realize a rendezvous with both stationary and moving targets from a resting state, a series of trajectory optimizations were performed. A point to point motion and a rendezvous task were both studied, using a single shooting method and direct collocation for point to point motion, and direct collocation for the rendezvous event.[5] Both the shooting method and direct collocation pose the trajectory problem as a boundary value problem, with objective functions guiding the optimization to achieve the state transition.

A. Point to point motion

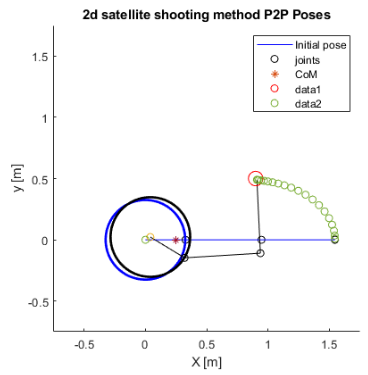
The point to point maneuver highlights the system capability to relocate the end effector to a specific location in the work space, ending with zero system velocity. The end point was chosen by proximity to an open-loop joint acceleration location to target a feasible point in the workspace.

1) shooting method: The criteria used for the point to point shooting method is shown in table II. The enforcement of the end state used inequality constraints to improve solution stability. The results of the optimization can be seen in figure 6.

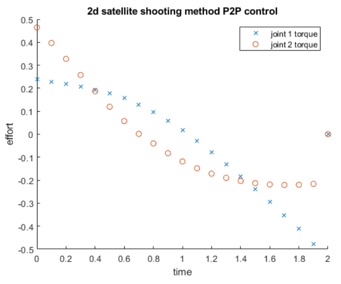
Criteria	value	unit
$EE_{pos,0}$	[0 1.54]	m
$EE_{pos,F}$	[0.9 0.5]	m
$EE_{vel,0}$	[0 0]	m/s
$EE_{vel,F}$	[0 0]	m/s
Time	2	seconds
Control bounds	± 2	Nm
End-state tolerance	0.01	m - m/s
cost function	control^2	-

TABLE II: Shooting method criteria: point 2 point

2) Direct Collocation: The same point to point maneuver was performed using direct collocation, using the criteria in table III. The output of the optimization is shown in figure 7, showing the end effector motion, initial and final pose, and control torques. The two methods offer substantially similar results, as can be seen in figures 6 and 7.

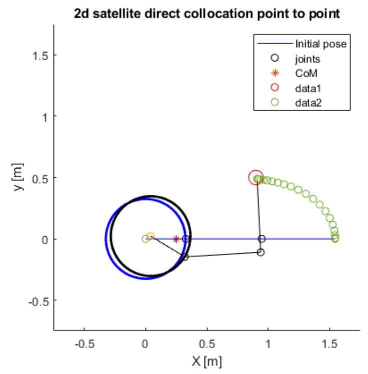


(a) Initial to final pose

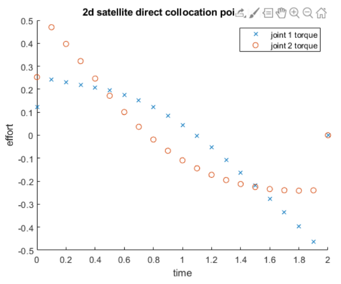


(b) Control torques

Fig. 6: Shooting method point to point



(a) Initial to final pose



(b) Control torques

Fig. 7: Direct Collocation point to point

Criteria	value	unit
$EE_{pos,0}$	[0 1.54]	m
$EE_{pos,F}$	[0.9 0.5]	m
$EE_{vel,0}$	[0 0]	m/s
$EE_{vel,F}$	[0 0]	m/s
Time	2	seconds
Control bounds	$\pm -$	Nm
End-state tolerance	0.01	m - m/s
Knot points	21	-
Collocation method	Trapezoidal	-
cost function	control^2	-

TABLE III: Direct collocation criteria: Point to point

B. *Rendezvous with moving point*

At the heart of the the related work [8] is the rendezvous maneuver. Full knowledge of the target body is assumed, and a the rendezvous time is specified, which poses the problem as a specified end effector position, velocity, and time of arrival. The satellite body is assumed to be starting from rest at a specified initial configuration. Direct collocation was used to perform the optimization, with criteria in Table IV.

Criteria	value	unit
$EE_{pos,0}$	[0.9 0.9]	m
$EE_{pos,F}$	[1.00 0.53]	m
$EE_{vel,0}$	[0 0]	m/s
$EE_{vel,F}$	[-0.110 -0.022]	m/s
Time	4	seconds
Control bounds	± 5	Nm
End-state tolerance	0.01	m - m/s
Knot points	21	-
Collocation method	Trapezoidal	-
cost function	control^2	-

TABLE IV: Direct collocation criteria: Point to point

The outputs of the optimization can be seen in figure 8. Figure 8b shows the optimized end effector trajectory intersecting target body, and figure 8d shows the position and velocity states aligning with the target body position and velocity at the specified rendezvous time.

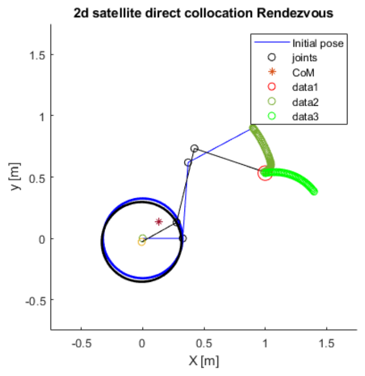
V. Controllers - Trajectory Tracking

A. *Comparison of MPC methods*

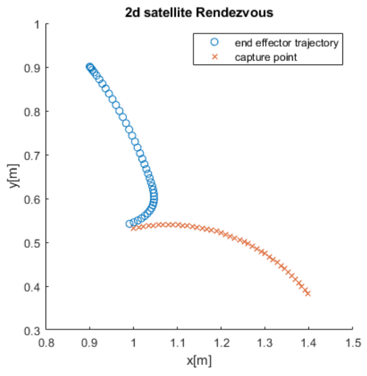
Parameter	Value
Mass, m	1 kg
Damping coefficient, b	0.3
Gravity, g	9.8 $\frac{m}{s^2}$
Length, l	1 m m
Prediction horizon	10 steps
Control horizon	2 steps
Control signal amplitude	10 $N-m$
Control signal frequency	1 Hz

TABLE V: Simulation parameters used for NMPC vs Adaptive MPC comparison

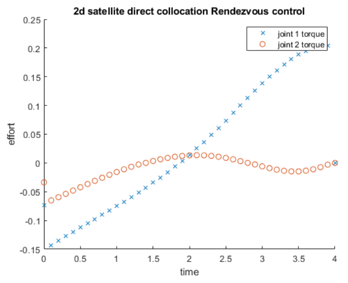
This section highlights a brief comparison of full non-linear MPC with adaptive MPC. We implemented both



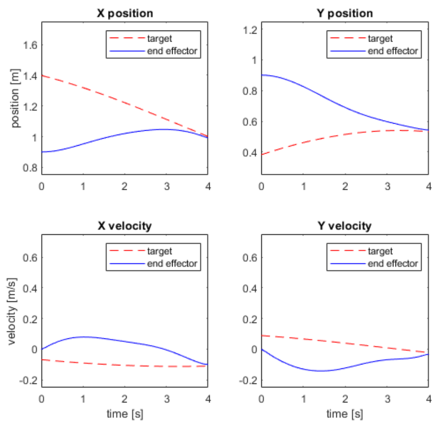
(a) Initial to final pose



(b) End effector trajectory



(c) Control torques



(d) End effector vs target point

Fig. 8: Direct Collocation rendezvous

NMPC and adaptive MPC using the Matlab MPC toolbox on a simple pendulum with parameters described by Table [V]. Table [**??**] shows the amount of time each method took to solve the optimal control problem. For this comparison, we defined a nominal control signal, ran it open loop to create a nominal state trajectory, and then used these nominal signals as reference for NMPC and adaptive MPC to track. For the simple pendulum experiments, we passed in two separate nominal trajectories: one sinusoidal signal with a small amplitude, and another sinusoidal control signal with a large amplitude. As can be seen in Table [**??**], NMPC takes about an order of magnitude longer to solve than adaptive MPC. The results of the tracking for NMPC and adaptive MPC can be found in figs 9 and 10. Clearly, in this case for the scenario with a control signal amplitude of 10 $N-m$, we can see that NMPC tracks the reference trajectory about as accurately as adaptive MPC. In this scenario, where the optimizer has access to a perfect model of the system dynamics, adaptive MPC seems to be the winner since it achieves similar performance, with near-real time computation speed.

We have noticed that the real trade-off between NMPC and adaptive MPC comes into play when the optimizer does not have access to a perfect system model. Table [VIII] shows the discrepancy between what the MPC optimizers were given versus what the actual dynamic properties were. When the optimizer does not have access to a perfectly accurate model of the system dynamics, NMPC out performs adaptive MPC on trajectory tracking, however this comes at the cost of require an extra order of magnitude of run-time. Table [VI] shows a comparison of adaptive MPC and full NMPC with and without model discrepancies.

Fig	Scenario	Run time	Avg error
9	NMPC - No model diff	21.41 s	0.0054 m
11	NMPC - Model diff	20.20 s	0.0091 m
12	Adaptive MPC - No model diff	2.48 s	0.003 rad
13	Adaptive MPC - Model diff	2.42 s	0.044 rad

TABLE VI: Simple Pendulum simulation comparison - NMPC vs Adaptive MPC (table IX - model differences)

Parameter	Value
Mass, m	1 kg
Damping coefficient, b	0.3
Gravity, g	9.8 $\frac{m}{s^2}$
Length, l	1 m m
Prediction horizon	10 steps
Control horizon	2 steps
Control signal amplitude	1 $N-m$
Control signal frequency	1 Hz

TABLE VII: Simulation parameters used for NMPC vs Adaptive MPC comparison (small amplitude)

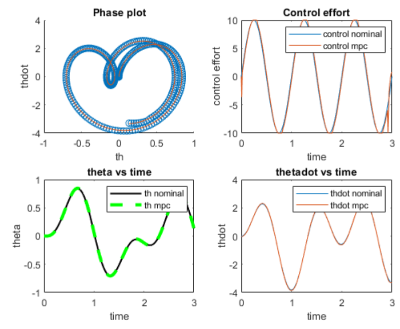


Fig. 9: Full NMPC Simple Pendulum Performance (large amplitude)

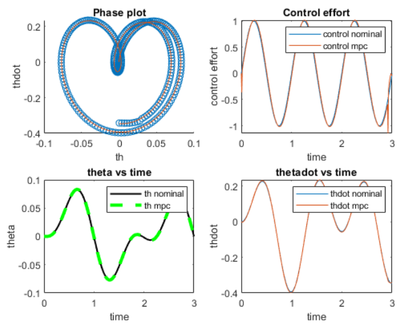


Fig. 10: Full NMPC Simple Pendulum Performance (small amplitude)

B. Rendezvous Maneuver with MPC

The reference paper based the realization of the planned trajectory around a non-linear MPC controller. The present work offers a comparison of adaptive MPC and non-linear MPC as related approaches to implementing the trajectory on the satellite model. The seeded trajectory is taken from the output of section IV-B. The target end effector positions and velocities serve as the desired propagated state. The dynamic system described

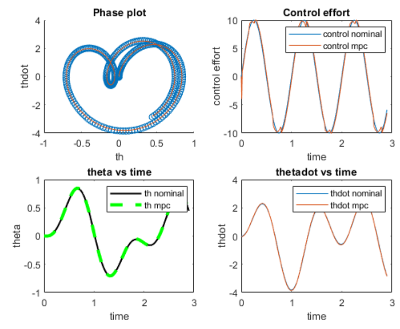
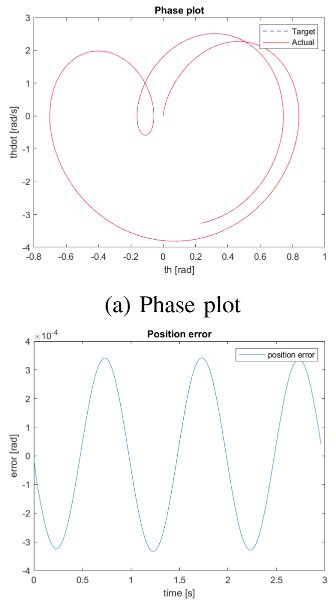
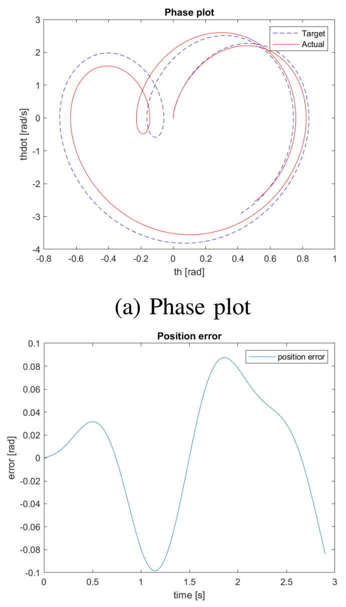


Fig. 11: Full NMPC Simple Pendulum Performance (discrepancy)


(a) Phase plot

(b) Position error avg 0.003, runtime 2.48

Fig. 12: Simple pendulum adaptive MPC with no model discrepancies


(a) Phase plot

(b) Position error avg 0.044, runtime 2.42

Fig. 13: Simple pendulum adaptive MPC with model discrepancies in table

Parameter	Value
Mass (given to optimizer), m	1 kg
Mass (real), m	0.9 kg
Damping coefficient, b	0.3
Gravity, g	9.8 $\frac{m}{s^2}$
Length (given to optimizer), l	1 m m
Length (real), l	1.1 m m
Prediction horizon	10 steps
Control horizon	2 steps
Control signal amplitude	10 $N-m$
Control signal frequency	1 Hz

TABLE VIII: Simulation parameters used for NMPC vs Adaptive MPC comparison (mass/length discrepancies)

in section III-A5 with the augmented state was used to perform the target maneuver, and model predictive controllers were implemented using the MATLAB MPC toolbox. The controllers are passed the state trajectory without the control sequence. This is deemed reasonable as the control is negligibly penalized in the cost function for the presented cases. Furthermore, the performance of the system following a desired trajectory without corresponding control information allows for a more general case of trajectory following. A further investigation would benefit from incorporation of the trajectory optimized control information to the control as a feed-forward term.

The maneuver was simulated with two different scenarios:

1) coherence between the internal model parameters used for the optimization and actual system parameters (no differences)
2) Model discrepancies between component masses

In the latter scenario, the trajectories are planned about the model, the controllers perform optimization on the model, and the control sequences are applied to the actual system to iterate the state. Table IX details the applied deviations between the models.

The outputs of the two different MPC methods applied to the rendezvous maneuver for both scenarios are summarized in table XI. The performance of the controllers is compared with total run time and average error. Comparison with respect to run time shows Adaptive MPC outperforming the non-linear controller by an order of magnitude. This improvement is expected due the optimization benefits from linearization of the system dynamics, as detailed in section III-B.

Comparing output performance in end effector error is more nuanced. Given an alignment of model and actual system parameters, adaptive MPC and non-linear MPC both offer similar state performance. However, introducing model inaccuracies significantly degrades the performance of adaptive MPC as compared to NMPC, which demonstrates a higher degree of robustness to system parameter inaccuracies. Again this difference in performance is expected, as a linearization further introduces prediction error into the optimization process. The relationship between model inaccuracies and controller performance for the satellite system mirrors the results from the controllers applied to the simple pendulums. Given the significant model deviations and an abrupt shift in target motion near the end of the maneuver, the adaptive controller fails to provide the necessary adjustments to maintain target proximity.

This set of test conditions highlight particular trade-offs between adaptive and non-linear MPC. Extending the study would entail further refinement of cost functions, optimization of run time code to remove specific implementation discrepancies, and adjustment of other parameters such as the control and prediction horizons to investigate controller sensitivities.

Component	Model deviation
Base	+30% mass, Inertia
Link 1	-30% mass, Inertia
Link 2	-30% mass, Inertia

TABLE IX: Satellite component deviations from real to model

Parameter	value	unit
prediction horizon	6	m
control horizon	2	m
Sample time	0.1	s
Control bounds	± 100	Nm
Cost	state^2	-

TABLE X: Direct collocation criteria: Point to point

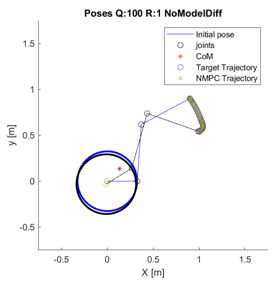
Fig	Scenario	Run time	Avg error
14	NMPC - No model diff	493.10 s	0.0101 m
15	NMPC - Model diff	568.54 s	0.0126 m
16	Adaptive MPC - No model diff	63.41 s	0.0090 m
17	Adaptive MPC - Model diff	63.64 s	0.059 m

TABLE XI: Simple Pendulum comparison - NMPC vs Adaptive MPC

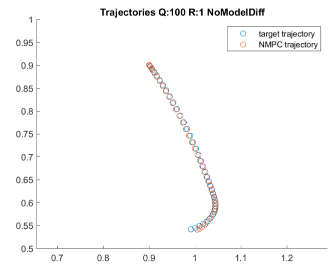
C. *Inverse dynamic control*

For the purpose of having another method for comparison, we also implemented an inverse dynamics controller for the purpose of tracking workspace trajectories for the 2 link planar space robot. We implemented the inverse dynamics controller by using the mass and Coriolis matrices given by SPART, along with the system's current state, and a computed desired joint space acceleration. Using all of these parameters, we were able solve for the necessary torque required to produced our desired joint space accelerations:

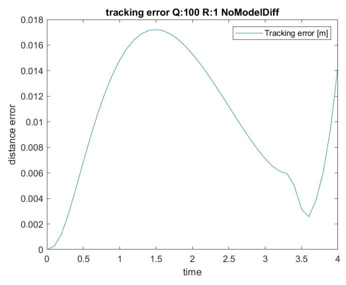
$$\tau = S^{-1}(MJ^{-1}(\dot{V}_{des} - \dot{J}\dot{q}) + C\dot{q} + N)$$



(a) Initial and final pose

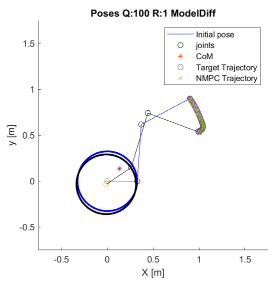


(b) End effector target vs realized

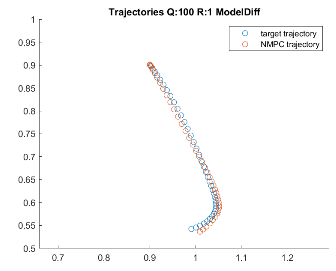


(c) Position error

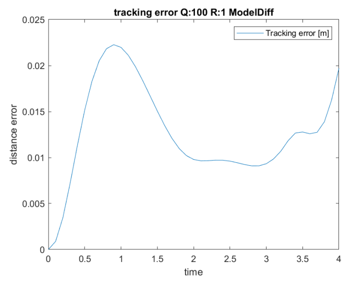
Fig. 14: Satellite rendezvous with Nonlinear MPC, no model differences



(a) Initial and final pose



(b) End effector target vs realized



(c) Position error

Fig. 15: Satellite rendezvous with Nonlinear MPC, model differences from table IX

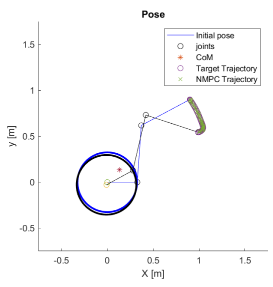
Where, $\dot{V}_{des}$ is the desired workspace acceleration, $\dot{J}$ is the time derivative of the system generalized Jacobian, and S is the selection matrix which ensures that the generalized forces applied to the base are unforced. We computed the desired workspace acceleration by wrapping a PD controller around the current end effector position and the desired end effector position defined by the reference trajectory. As can be seen in Figs 11 and 12, the inverse dynamics controller does not perform quite as well at the trajectory tracking task as the NMPC controller we implemented. Further work should be done tuning the PD gains on the inverse dynamics controller to verify the findings.

VI. Conclusions and Discussion

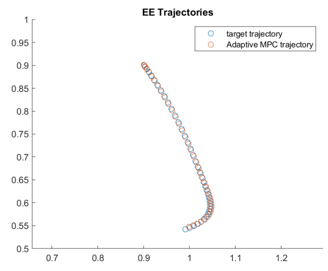
In conclusion, we presented our implementation and extension of Tomasz Rybus's 2017 work on nonlinear model predictive control for a planar space robot. We went into the background of the trajectory optimization and dynamics of the system and then explained the details of our method. We took his work one step further by examining the performance and efficacy of using adaptive MPC instead of full NMPC for the space robot system as well as for a simple pendulum system. Through the scenarios presented, adaptive MPC performed significantly faster with similar tracking performance to that of full NMPC – when adaptive MPC has access to a perfect dynamics model. However, when adaptive MPC does not have access to a perfect dynamics model, for example, if the masses or geometry are wrong, NMPC's tracking capabilities are more robust to these discrepancies in dynamic parameters. All code from the project can be found at the following GitHub page: GitHub

Further investigation of methods presented would benefit from the following.

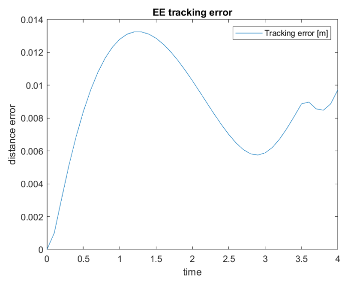
- Soft constraints - An easement of the constraints imparted to the optimization problem allows for



(a) Initial and final pose

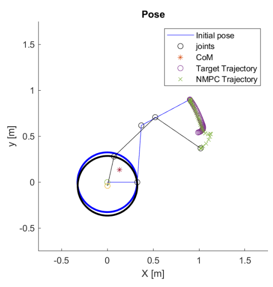


(b) End effector target vs realized

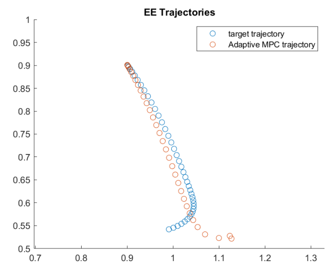


(c) Position error

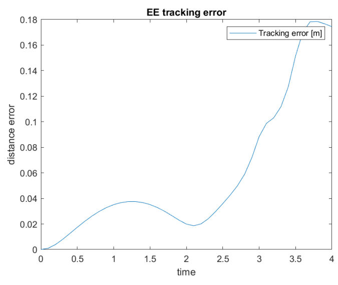
Fig. 16: Satellite rendezvous with Adaptive MPC, no model differences



(a) Initial and final pose



(b) End effector target vs realized



(c) Position error

Fig. 17: Satellite rendezvous with Adaptive MPC, model differences from table IX

minor violations of the constraints, managed by a constraint violation cost component. This adaptation of the constraint conditions can improve the robustness of feasible solutions, as the constraint relaxation provides more leniency in the solution search.

- Spatial system - 3D operation: Future work would feature implementations of adaptive MPC and NMPC on a 3D system.
- Minimum time to capture: Future work would also feature an analysis into minimum capture time problems - that is, how can the space robot plan a trajectory to perform a capture operation in minimum time. This is slightly different (and harder) than the rendezvous problem considered in this work, because in this work we 'hard coded' the end time of the maneuver. In a minimum time problem, the optimizer would have to search for the quickest possible end time, instead of having such a parameter passed in already.

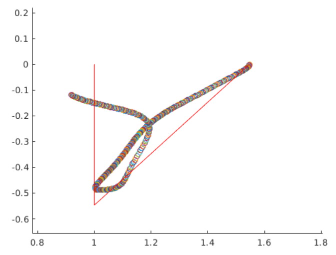
References

[1] MATLAB adaptive robust numerical differentiation. https://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation, . Accessed: 2020-12-16.

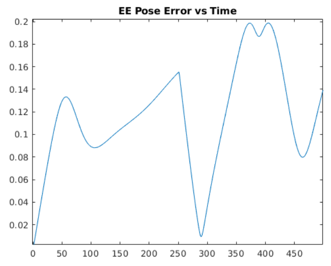
[2] MATLAB model predictive control. https://www.mathworks.com/help/mpc/index.html?s_tid=CRUX_lftnav, . Accessed: 2020-12-16.

[3] MATLAB optimization problem. https://www.mathworks.com/help/mpc/ug/optimization-problem.html, . Accessed: 2020-12-16.

[4] MATLAB constrained nonlinear opimization algorithms. https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#f26684, . Accessed: 2020-12-16.

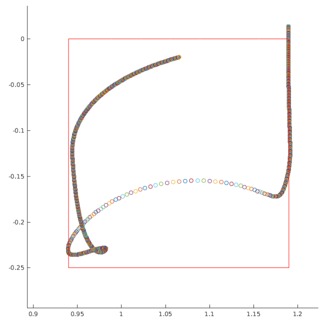


(a) EE position relative to reference traj

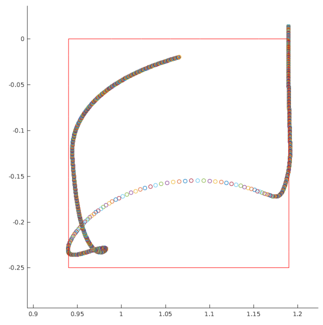


(b) Error history for acute angle trajectory

Fig. 18: Inverse Dynamics for acute angle trajectory tracking



(a) EE position relative to reference traj



(b) Error history for square trajectory

Fig. 19: Inverse Dynamics for square trajectory tracking

[5] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation.

[6] Mitsushige Oda, Kouichi Kibe, and Fumio Yamagata. Ets-vii, space robot in-orbit experiment satellite. In *Proceedings of IEEE international conference on robotics and automation*, volume 1, pages 739–744. IEEE, 1996.

[7] Andrew Packard, Roberto Horowitz, Kameshwar Poolla, and Francesco Borrelli. ME 132, Dynamic Systems and Feedback. page 493, 2018.

[8] Tomasz Rybus, Karol Seweryn, and J. Sasiadek. Control system for free-floating space manipulator based on nonlinear model predictive control (nmpc). *Journal of Intelligent Robotic Systems*, 85, 03 2017. doi: 10.1007/s10846-016-0396-2.

[9] Josep Virgili-Llop et al. SPART: an open-source modeling and control toolkit for mobile-base robotic multibody systems with kinematic tree topologies. https://github.com/NPS-SRL/SPART.

[10] CG Wagner-Bartak, JA Middleton, and JA Hunter. Shuttle remote manipulator system hardware test facility. In *11th Space Simulation Conference*, volume 2150, page 79, 1980.