Nonlinear Model Predictive Control for Hydrobatics: Experiments with an Underactuated AUV

Sriharsha Bhat, *Chariklia Panteli*, *Ivan Stenius*, *and Dimos V. Dimarogonas* *

Abstract

Hydrobatic Autonomous Underwater Vehicles (AUVs) can be efficient in range and speed, as well as agile in maneuvering. They can be beneficial in scenarios such as obstacle-avoidance, inspections, docking, and under-ice operations. However, such AUVs are underactuated systems - this means exploiting the system dynamics is key to achieving elegant hydrobatic maneuvers with minimum controls. This paper explores the use of Model Predictive Control (MPC) techniques to control underactuated AUVs in hydrobatic maneuvers and presents new simulation and experimental results with the small and hydrobatic SAM AUV. Simulations are performed using nonlinear MPC (NMPC) on the full AUV system to provide optimal control policies for several hydrobatic maneuvers in Matlab/Simulink. For implementation on AUV hardware in ROS, a linear time varying MPC (LTV-MPC) is derived from the nonlinear model to enable real-time control. In simulations, NMPC and LTV-MPC shows promising results to offer much more efficient control strategies than what can be obtained with PID and LQR based controllers in terms of rise-time, overshoot, steady-state error and robustness. The LTV-MPC shows satisfactory real-time performance in experimental validation. The paper further also demonstrates experimentally that LTV-MPC can be run real-time on the AUV in performing hydrobatic maneouvers.

Keywords: Underactuated robots, optimization and optimal control, marine robotics, field testing, autonomous underwater vehicles, model predictive control, nonlinear systems, simulation.

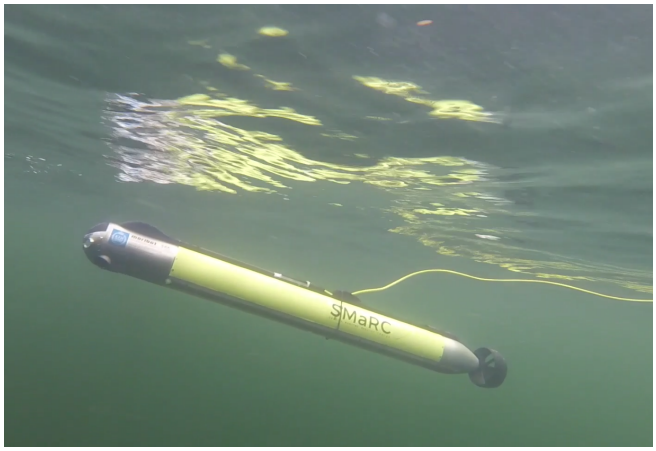


Fig. 1: The hydrobatic SAM AUV in field tests in Kristineberg, Swedish west coast

1 Introduction

The term hydrobatics stems from aerobatics and refers to agile maneuvering of underwater robots. Hydrobatic Autonomous Underwater Vehicles (AUVs) can be efficient in terms of range and speed, as well as agile in maneuvering. This combination of efficiency and agility can enable new capabilities for use cases of AUVs in ocean production, environmental sensing and security [Bhat and Stenius, 2018]. These capabilities can include increased perception in unknown environments [Bosch et al., 2015], precise positioning and docking with moving targets [Berg and Wicklander, 2016, Wilson, 2009], and collaboration in a swarm [Özkahraman and Ögren, 2020]. In practice, such AUVs are underactuated (and have saturation limits), can have highly nonlinear dynamics (especially during hydrobatic maneuvers at high incidence angles), and operate in a dynamic and partially observed environment.

A key benefit of hydrobatic AUVs in field deployments is that they can effectively perform transits and longer missions, while being agile in critical scenarios such as obstacle avoidance, escape from pursuit, inspection of targets in confined spaces, near/under ice launch and recovery, and vertical hovering for station-keeping or sampling. In these scenarios, hydrobatic maneuvers such as helical loops (e.g. for panoramic inspections), tight turns to avoid obstacles, and precise positioning (e.g. for docking) place high requirements on sensing, actuation and modeling. This is due to measurement uncertainty, underactuation, saturation limits and unsteady flow conditions. To be successful in these critical scenarios, a key challenge is to control an underactuated AUV system and find optimal control sequences to efficiently perform hydrobatic maneuvers [Bhat and Stenius, 2018, Duecker et al., 2021]. By addressing this challenge, simple, cost effective and agile robot configurations could yet offer high performance in autonomy and endurance.

Underactuated robotics is an active research area,

*This work was supported by the Swedish Foundation for Strategic Research (SSF) through the Swedish Maritime Robotics Center (SMaRC). S.Bhat (svbhat@kth.se, corresponding author) and I.Stenius (stenius@kth.se) are with the School of Engineering Sciences, KTH Royal Institute of Technology, Stockholm, Sweden. D.V.Dimarogonas (dimos@kth.se) is with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. C.Panteli (charikliapanteli@gmail.com) was at KTH Royal Institute of Technology when the work for this paper was performed, and is now with Cleeven Sc, Stockholm, Sweden.

with significant interest in motion planning, control, and learning. Several publications focus on agile robot maneuvering tasks and employ methods including classical PID and backstepping control [Bulka and Nahon, 2018, Ferreira et al., 2012], feedback motion planning [Majumdar and Tedrake, 2017, Tedrake et al., 2010, Cory and Tedrake, , Moore and Tedrake, 2012], supervised learning [Kaufmann et al., 2020], reinforcement learning [Abbeel et al., 2007], optimal control [Mueller and D'Andrea, 2013] and model predictive control [Bangura and Mahony, 2014, Ru and Subbarao, 2017, Gros et al., 2012].

Model predictive control, in particular, offers a pragmatic choice in underactuated robotics. The use of a finite prediction horizon can enable real-time implementations, since a locally optimal control can be recalculated periodically. Augmentations can also be made to guarantee infinite horizon stability and convergence [Chen and Allgöwer, 1997]. The entire system dynamics model can be used during the computation, thereby exploiting couplings with uncontrolled states [Mayne et al., 2000]. The ability to set constraints on states and controls enables us to account for saturation limits, and even account for model and measurement uncertainties (e.g. using tube-like state constraints [Mayne et al., 2011, Lu and Cannon, 2019]). MPC methods can also be closely linked to learning - by offering expert demonstrations [Kaufmann et al., 2020], using learned dynamics models [Amos et al., 2018, Berberich et al., 2020, Lorenzen et al., 2019], and providing stability guarantees [Zhang et al., 2016, Fan et al., 2020]. However, as with other optimal control methods, a key limitation is that the control policy generated is only as good as the model and solver used, which means modeling and implementation is crucial.

MPC implementations on autonomous underwater vehicles include trajectory planning [Shen et al., 2015], trajectory tracking [Shen et al., 2016, Shen et al., 2019, Yan et al., 2020], formation control[Gomes and Pereira, 2018, Gomes and Pereira, 2019], manipulation [Nikou et al., 2018] and docking [Nielsen et al., 2018]. Most of these developments have been demonstrated in simulation studies, and very few cases of experimental and field demonstrations exist. Notably, in [Steenson et al., 2014], MPC is used for speed and depth control of a hybrid AUV in field conditions. Additionally, experimental nonlinear MPC studies are presented in [Saback et al., 2020] for velocity control and in [Heshmati-alamdari et al., 2018] for waypoint tracking in a constrained environment respectively. Two broad reasons exist for the small number of field and experimental implementations. First, the dimension of the optimization problem can be too large for real-time control if the full dynamics model (with over 12 states) is used. This computational issue can be addressed by combining reduced order models of weakly interacting subsystems [Shen et al., 2016], or by generating pre-computed lookup tables of attainable controls and trajectories [Gomes and Pereira, 2018]. Second, uncertainty in state measurement and hydrodynamic modelling is high due to sensing constraints (low visibility, limited localization, lack of GPS) and the nature of the underwater environment (unsteady flow, currents, disturbances). Such uncertainty can be addressed with good state estimation [Steenson et al., 2014], with additional safety and robustness constraints [Nikou et al., 2021], or by extending the MPC to hybrid automata (including state machines or behavior trees) [Gomes and Pereira, 2019]. To the best of our knowledge, there have been no real-time MPC implementations for agile underactuated AUVs.

This paper presents a method to control underactuated AUVs in hydrobatic maneuvers online in real time using model predictive control (MPC), together with new simulation and experimental results. The following contributions are made:

1. A nonlinear MPC for underactuated AUVs is presented. For real-time control, the nonlinear MPC is linearized periodically to obtain a Linear Time-Varying MPC (LTV-MPC). Both these controllers are applied to perform hydrobatic maneuvers with an AUV model.

2. The nonlinear MPC is validated in Simulink within a high-fidelity simulation environment. It provides better performance and robustness to disturbances than the baseline controllers.

3. The LTV-MPC is implemented and tested with field experiments on the agile and hydrobatic SAM AUV (using the CVX convex optimization library). SAM, short for Small and Affordable Maritime Robot, is a 1.5m long torpedo shaped research AUV platform developed by the Swedish Maritime Robotics Center ([Bhat et al., 2019], see Fig. 1). This is one of the first real-world demonstrations of hydrobatic capabilities using MPC on AUV hardware.

The subsequent sections will show that using MPC can enable efficient and elegant control strategies for complex maneuvers. Furthermore, implementations on robot hardware can validate the use of MPC for underactuated AUVs.

2 Preliminaries and problem formulation

2.1 Model Predictive Control

In optimal control, the control problem is reformulated as an optimization problem with an objective and constraints. Model predictive control offers a pragmatic approach to solve the following optimal control problem online in real time. The open loop optimal control problem is solved for a finite (relatively short) prediction horizon T_p, and the optimal control is applied for only the first time step (or a pre-defined control horizon). In continuous time, such a problem formulation can be presented as:

$$\min_{u} \quad J = \phi(x_p) + \int_t^{t+T_p} f_0(x,u)\mathrm{d}t,$$

$$\text{s.t} \quad \begin{aligned} \dot{x} &= f_1(x,u), \\ x &\in \mathbb{X}, \\ u &\in \mathbb{U}, \end{aligned} \tag{1}$$

where x represents the states, u represents the control input, $\phi(x)$ is the cost at the final state at the prediction horizon $x(T_p)$, t and $t+T_p$ refer to the current time and the final time after the prediction horizon respectively while f_0 is the objective function at each time instant. The dynamics of the system are represented by a nonlinear ordinary differential equation (ODE) through the function $f_1(x,u)$. The states and controls are constrained to sets $\mathbb{X}$ and $\mathbb{U}$.

At each time step, the optimal control u^* is recalculated and the states are updated. This receding horizon strategy enables fast computations (due to the smaller horizon) while adding some robustness to disturbances (due to recalculation of the optimal control). In most cases, the continuous time problem formulation is discretized, enabling the use of numerical optimization solvers.

In this work, we employ nonlinear model predictive control to solve the optimal control problem, by using techniques from convex optimization and nonlinear programming to obtain optimal solutions at each time step.

2.2 Dynamics model

A hydrobatic AUV (Fig. 2) is modeled with quaternion kinematics and nonlinear dynamics using the notation given by Fossen [Fossen, 2011]. An NED coordinate system is used to represent the world frame (with a positive downwards Z_E axis), while a right handed Cartesian coordinate system is used in the AUV frame of reference. The kinematics in 6DOF are represented by the relation

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J_q}(\boldsymbol{\eta})\boldsymbol{\nu}\ , \tag{2}$$

where $\boldsymbol{J_q} \in \mathbb{R}^{7\times 6}$ is a combined transformation matrix between the pose vector $\boldsymbol{\eta}$ and the velocity vector $\boldsymbol{\nu}$. The pose vector $\boldsymbol{\eta} = \begin{bmatrix} x_E & y_E & z_E & \varepsilon_0 & \varepsilon_1 & \varepsilon_2 & \varepsilon_3 \end{bmatrix}^T$, contains the positions and quaternion orientations in the world frame. Unit quaternions are used instead of Euler angles in order to avoid a singularity at 90° pitch (see e.g. [Silva and Sousa, 2008]) during hydrobatic maneuvers. The velocity vector $\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T$ contains the translational and rotational velocities with respect to the x_B, y_B and z_B axes in the body fixed frame.

A vectorial robot-like representation, as presented by Fossen [Fossen, 2011], is used to describe the dynamics of the AUV as follows:

$$\begin{aligned} (\boldsymbol{M_{RB}} + \boldsymbol{M_A})\dot{\boldsymbol{\nu}} + (\boldsymbol{C_{RB}}(\boldsymbol{\nu}) + \boldsymbol{C_A}(\boldsymbol{\nu}))\boldsymbol{\nu} + \\ \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + g(\boldsymbol{\eta}) = \boldsymbol{\tau_C}\ , \end{aligned} \tag{3}$$

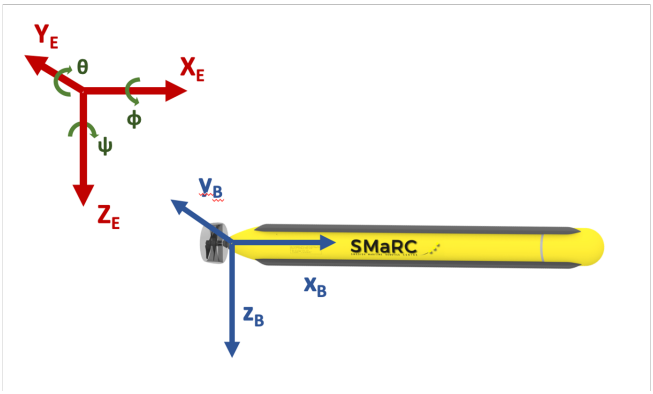


Fig. 2: AUV world and body reference frames and angle definitions.

where $\boldsymbol{M_{RB}}$ is the rigid body mass and inertia matrix and $\boldsymbol{C_{RB}}$ is the matrix of Coriolis and centripetal terms on the left hand side. $\boldsymbol{M_A}$ and $\boldsymbol{C_A}(\nu)$ represent the effect of added mass, $\boldsymbol{D}(\nu)$ represents the damping matrix and $\boldsymbol{g}(\eta)$ is the vector of gravitational and buoyancy forces and moments. $\boldsymbol{\tau_C}$ is a vector of external control forces, and depends on the robot's actuator configuration.

Lumping together the mass terms and the damping terms in (3) as effective mass and damping matrices, $\boldsymbol{M}_{\text{eff}} = \boldsymbol{M_{RB}} + \boldsymbol{M_A}$ and $\boldsymbol{C}_{\text{eff}}(\boldsymbol{\nu}) = \boldsymbol{C_{RB}}(\boldsymbol{\nu}) + \boldsymbol{C_A}(\boldsymbol{\nu}) + \boldsymbol{D}(\boldsymbol{\nu})$, we get:

$$\boldsymbol{M}_{\text{eff}}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{\text{eff}}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau_C}\ , \tag{4}$$

Equation (4) can now be rewritten as:

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}_{\text{eff}}^{-1}(\boldsymbol{\tau_C} - \boldsymbol{C}_{\text{eff}}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{g}(\boldsymbol{\eta}))\ . \tag{5}$$

Equations (2) and (5) combine to provide the nonlinear dynamics model $d(x,u)$ in (1) with the state vector $\begin{bmatrix} \boldsymbol{\eta} & \boldsymbol{\nu} \end{bmatrix}^T$ and control vector $\boldsymbol{\tau_C}$. The quaternion orientations in the state vector $\boldsymbol{s^T} = \begin{bmatrix} \boldsymbol{\eta} & \boldsymbol{\nu} \end{bmatrix}^T$ are converted to Euler angles by an output function $\boldsymbol{o}(\boldsymbol{\eta}, \boldsymbol{\nu})$.

2.3 Problem formulation

In this paper, nonlinear model predictive control (NMPC) is considered for output reference tracking (where the optimal control must minimise the deviation of the output from a reference value) with the following problem formulation:

Problem 1: Consider an AUV with its output state given by $\boldsymbol{s}^{\boldsymbol{T}}_{\text{out}} = \begin{bmatrix} \boldsymbol{\eta}_{\text{out}} & \boldsymbol{\nu}_{\text{out}} \end{bmatrix}^T$, and the reference state to be tracked given by $\boldsymbol{s}^{\boldsymbol{T}}_{\text{ref}} = \begin{bmatrix} \boldsymbol{\eta}_{\text{ref}} & \boldsymbol{\nu}_{\text{ref}} \end{bmatrix}^T$. We assume the vector of control forces $\boldsymbol{\tau_C}$ to be a function of the vector of available control inputs $\boldsymbol{c}$. The reference tracking problem is then formulated as

$$
\begin{aligned}
\min_{\boldsymbol{\tau_C}} \quad & J = \int_t^{t+T_p} [(\boldsymbol{s}_{\text{ref}} - \boldsymbol{s}_{\text{out}})^T \boldsymbol{Q} (\boldsymbol{s}_{\text{ref}} - \boldsymbol{s}_{\text{out}}) + \\
& \boldsymbol{c}^T \boldsymbol{R_1} \boldsymbol{c} + \dot{\boldsymbol{c}}^T \boldsymbol{R_2} \dot{\boldsymbol{c}}] \mathrm{d}t \\
s.t. \quad & \\
& \boldsymbol{s}_{\text{out}} = \boldsymbol{o}(\boldsymbol{\eta}, \boldsymbol{\nu}), \\
& \dot{\boldsymbol{\eta}} = \boldsymbol{J_q}(\boldsymbol{\eta}) \boldsymbol{\nu} \ , \\
& \dot{\boldsymbol{\nu}} = \boldsymbol{M}_{\text{eff}}^{-1} (\boldsymbol{\tau_C}(\boldsymbol{c}) - \boldsymbol{C}_{\text{eff}}(\boldsymbol{\nu}) \boldsymbol{\nu} - \boldsymbol{g}(\boldsymbol{\eta}) \ , \\
& \boldsymbol{s}_{\text{out}} \in \mathbb{S}, \\
& \boldsymbol{\tau_C} \in \mathbb{T}.
\end{aligned}
\tag{6}
$$

where t is the time, T_p is the prediction horizon, and $\boldsymbol{Q}, \boldsymbol{R_1}$ and $\boldsymbol{R_2}$ are weighting matrices. The state dynamics are given by equations (2) and (5), and are subject to state and control constraints contained in sets $\mathbb{S}$ and $\mathbb{T}$.

The NMPC aims to minimize the deviation from the reference trajectory, while also minimizing the control input $\boldsymbol{c}$ and its rate $\dot{\boldsymbol{c}}$. The formulation can be customized, and it is possible to include additional constraints, objectives, and allow model updates. The initial state is also constrained within the state constraint set which falls within the stabilizable region $\boldsymbol{s}_{\text{out}}(0) \in \mathbb{S}$. If the state constraints constrain the state within a region Ω, then with $\Omega \subseteq \mathbb{S}$, the formulation above can be a stabilizing controller. Furthermore, for practical implementations, if the terminal state of the system region is in a stabilizable region (as it is here), then a nominal controller can be used to keep the system neutral within the terminal set, thus reducing computational effort.

In order to solve Problem 1, the following strategies are considered:

1. **Nonlinear MPC (NMPC)** using sequential quadratic programming(SQP) [Betts, 2010]. Here the SQP algorithm in the Matlab optimization library [1] is used. Continuous time dynamics are discretized using zero-order hold for a specified sampling time.

2. **Linear Time Varying MPC (LTV-MPC)** using convex optimization. Here, the nonlinear system is periodically linearized with an appropriate resolution to reduce the computational complexity, and in addition to Matlab's SQP algorithm, the OSQP[2] algorithm in CVXPY[3] is used.

In the first strategy, the solver speed of the nonlinear solver could limit the use of MPC for online real-time implementations due to a large state space, especially if nonlinear dynamics in 6DOF is considered (as it is here). Linearizing the model around a specific reference state simplifies the optimization problem, and can enable rapid solutions. The second strategy can therefore enable real-time implementations. The nonlinear solution is compared to LTV solutions in simulation. The best-performing real-time LTV solution can then be used in the robot hardware.

[1] https://se.mathworks.com/help/mpc/ug/configure-optimization-solver-for-nonlinear-mpc.html
[2] https://osqp.org/
[3] https://www.cvxpy.org/

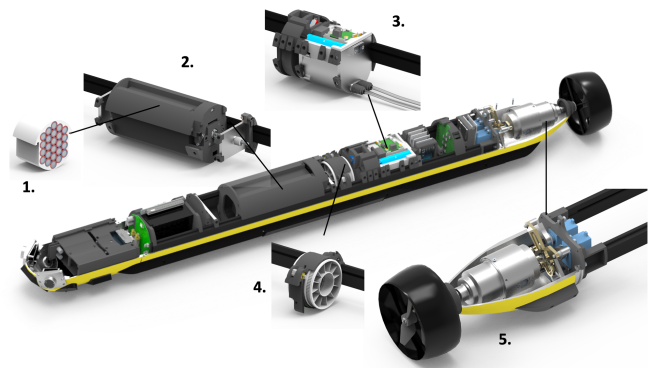


Fig. 3: SAM AUV subsystems: 1. Battery pack, 2. Longitudinal Center of Gravity Trim System (LCG), 3. Variable Buoyancy System (VBS), 4. Transversal Center of Gravity System (TCG), 5. Thrust Vectoring System with Counter-rotating Propellers.

3 Implementation

The presented method is implemented on the hydrobatic SAM AUV as a test case [Bhat et al., 2019, Bhat et al., 2020]. This section specifies implementation details including the nature of the system, and the control algorithms used.

3.1 The SAM AUV

SAM is a hydrobatic AUV, and has a unique actuator configuration to make it agile, while being underactuated (Fig. 3). Counter-rotating propellers are used for propulsion, while a thrust vectoring nozzle is used for maneuvering. A variable buoyancy system enables static depth control by pumping water in and out of a tank. A movable battery pack enables longitudinal changes to center of gravity (c.g.) position and allows pitch control, while rotating counterweights allow transversal c.g. changes and offer static roll control. With this actuator configuration, both static and dynamic changes to state are possible, and MPC offers exciting opportunities for controlling this system. Navigation and payload sensors include an IMU, a compass, a GPS (for surface operations), a Doppler Velocity Logger, cameras, sonar and environmental sensors. From a software perspective, the autonomy packages on SAM run on the Robot Operating System (ROS) environment, and subcomponents include mission planning, robust mission execution (using behavior trees), motion planning (using spline-fitting), feedback control (cascaded PID controllers) and dead reckoning (with an Extended Kalman Filter).

The actuator equations are integrated into the system dynamics model used for MPC in (5), with 6 control inputs, $\boldsymbol{c} = [\ \text{rpm}_1 \quad \text{rpm}_2 \quad \delta_e \quad \delta_r \quad \text{vbs} \quad \text{lcg}\ ]$. These are integrated as follows:

1. The thrust forces (F_T) and moments (M_T) are modeled as linear functions of the propeller rotational

rates (rpm) as

$$\begin{bmatrix} F_T \\ M_T \end{bmatrix} = \begin{bmatrix} K_F & K_F \\ K_M & K_M \end{bmatrix} \begin{bmatrix} \text{rpm}_1 \\ \text{rpm}_2 \end{bmatrix}, \tag{7}$$

where K_F and K_M are constant force and moment coefficients and $\text{rpm}_{1,2} \in [\text{rpm}_{1,2,\text{min}}; \text{rpm}_{1,2,\text{max}}]$.

2. The thrust vectoring systems align the thrust force in the right direction using effective elevator (δ_e) and rudder (δ_r) deflections. On rotating the thrust forces and moments with respect to δ_e and δ_r, the control force vector $\boldsymbol{\tau_C}$ is modeled as:

$$\boldsymbol{\tau_C} = \begin{bmatrix} F_T \cdot \cos\delta_e \cdot \cos\delta_r \\ -F_T \cdot \sin\delta_r \\ F_T \cdot \sin\delta_e \cdot \cos\delta_r \\ M_T \cdot \cos\delta_e \cdot \cos\delta_r \\ -M_T \cdot \sin\delta_r \\ M_T \cdot \sin\delta_e \cdot \cos\delta_r \end{bmatrix}, \tag{8}$$

where $\delta_{e,r} \in [\delta_{e,r,\text{min}}; \delta_{e,r,\text{max}}]$.

3. The longitudinal center of gravity system influences the $\boldsymbol{M}_{\text{eff}}$ and $\boldsymbol{C_{RB}}$ matrices by changing the c.g. x position as

$$x_g = x_{g0} + \text{lcg}, \tag{9}$$

where $\text{lcg} \in [\text{lcg}_{\text{min}}; \text{lcg}_{\text{max}}]$.

4. The variable buoyancy system is modeled as an additional buoyancy force that augments or counteracts the existing weight W to influence the net buoyancy as

$$\boldsymbol{g(\eta)} = W + \text{vbs}, \tag{10}$$

where $\text{vbs} \in [\text{vbs}_{\text{min}}; \text{vbs}_{\text{max}}]$.

3.2 MPC Algorithms

Two MPC algorithms based on the NMPC and LTV-MPC strategies respectively have been implemented for studying hydrobatic maneuvers with SAM. These discrete-time algorithms aim at driving the system to follow a reference trajectory, within a pre-defined error threshold ϵ. Note that in discrete time, we use the standard notations for states (x_k), outputs (y_k, y_{ref}) and controls (u_k) at each time-step k to represent their continuous time counterparts for states ($\boldsymbol{s}$), outputs ($\boldsymbol{s_{out}}$, $\boldsymbol{s_{ref}}$) and controls ($\boldsymbol{\tau_C}$) in equation (6). At each time instant k (with time step T_s), the algorithms take the current state x_k and the reference trajectory y_{ref} as input, and return the optimal control u_k as the output - u_k is applied to the system. When the system is within the error threshold ϵ, a nominal control (u_0) is applied to keep the AUV in a neutral state. This nominal control is computed when the AUV is first launched based on the steady actuator commands that keep it stationary and stable in its current weight and trim configuration (e.g. 0 *rpm* to the propellers, no steering, 50% VBS and LCG levels). When the system exits the threshold region, the MPC is reapplied. This aims at saving energy and computational power while improving stability.

Parameters for MPC include weights on states ($\boldsymbol{Q}$) and controls ($\boldsymbol{R_{1,2}}$), as well as the prediction horizon (T_p) and control horizon (m). Algorithm 1 describes the NMPC implementation, while Algorithm 2 presents the real-time LTV variant. Both algorithms are implemented in Matlab/Simulink, while Algorithm 2 is also implemented in Python/ROS.

In Algorithm 1, the nonlinear MPC presented in Problem 1 is solved using Sequential Quadratic Programming, and this solver is encapsulated in a function called *ComputeNMPC*. The state feedback is computed at each time-step, quaternions are converted to Euler angles for output tracking, and the optimal control is computed for a finite prediction horizon T_p considering the objective and constraints. The computed control is applied to the plant until the objective is achieved.

Algorithm 1: Nonlinear MPC, discrete time

Inputs: x_k, y_{ref}
Outputs: u_k
Parameters: $T_f, T_s, \boldsymbol{Q}, \boldsymbol{R_1}, \boldsymbol{R_2}, T_p, m$
for $k = 0$ **to** T_f/T_s **do**
  y_{ref} = GetRefTrajectory(k)
  x_k = GetStateFeedback(k)
  y_k = ConvertQuatToEul(x_k)
  u_k = ComputeNMPC(...
  Dynamics(x_k, u_k),$\text{y}_{\text{ref}}, y_k, T_p, m, \boldsymbol{Q}, \boldsymbol{R_1}, \boldsymbol{R_2}$)
  if $y_{ref} - y_k \geq$ *Threshold* **then**
    ApplyControl(u_k, m)
    else
      ApplyControl(u_0)
      Return Success
      Exit Loop
end

In Algorithm 2, a similar procedure is followed, but instead of the nonlinear model, a linearized dynamics model is used to compute the optimal control. The nonlinear system is approximated by a piecewise linear system, with a specified linearization resolution. For example, a coarse resolution would include linearization every 100 or 1000 time steps, while a fine resolution would consider linearization every 10 or 20 time steps. If the nonlinear dynamics are represented by a function $\dot{x} = f(x, u)$ with state x and control u, and the output function is presented as $y = o(x)$, then at each linearization step, Jacobian matrices are computed by calculating the gradient with respect to the relevant state/control vector. This means the linearized system in the neighborhood of a point $\tilde{x}, \tilde{u}$ can be obtained as $A_{\text{Jac}} = \nabla(f(x,u), \tilde{x})$, $B_{\text{Jac}} = \nabla(f(x,u), \tilde{u})$, $C_{\text{Jac}} = \nabla(o(x), \tilde{x})$. If this operation is performed periodically, then the linearized system at a time step k is represented by

$$\begin{aligned} \dot{x} &= A_{\text{Jac}}(k)\tilde{x} + B_{\text{Jac}}(k)\tilde{u}\,, \\ \tilde{y} &= C_{\text{Jac}}(k)\tilde{x}\,. \end{aligned} \tag{11}$$

This system is in a standard linear time-varying state space form and several solvers (including OSQP) can be used to solve the optimization problem.

Algorithm 2: Linearized Nonlinear MPC, discrete time

Inputs: x_k, y_{ref}
Outputs: u_k
Parameters: $T_f, Ts, \boldsymbol{Q}, \boldsymbol{R_1}, \boldsymbol{R_2}, T_p, m, Res$
for $k = 0$ **to** T_f/T_s **do**
  $y_{\text{ref}} = \text{GetRefTrajectory}(k)$
  $x_k = \text{GetStateFeedback}(k)$
  $y_k = \text{ConvertQuatToEul}(x_k)$
  $u_k = \text{ComputeMPC}(...$
  $[A_{\text{Jac}}, B_{\text{Jac}}, C_{\text{Jac}}], y_{\text{ref}}, y_k, x_k, T_p, m, \boldsymbol{Q}, \boldsymbol{R_1}, \boldsymbol{R_2})$
  if $k \mod Res == 0$ **then**
    $[A_{Jac}, B_{Jac}, C_{Jac}] =$
    $GetLinearModel(Dynamics, x_k, u_k, y_k)$
  if $y_{ref} - y_k \geq Threshold$ **then**
    $ApplyControl(u_k, m)$
    else
      $ApplyControl(u_0)$
      $Return\ Success$
      $Exit\ Loop$
  end

In the algorithms presented, the function *GetRefTrajectory* obtains the reference values for the states from a path planner, while *GetStateFeedback* obtains the current state feedback from the navigation and dead-reckoning software. ConvertQuatToEul transforms the orientation from quaternions to Euler angles for calculating the state error. *Dynamics* encapsulates the nonlinear dynamics model of the system based on equations (2) and (5), while *GetLinearModel* linearizes the nonlinear model at a particular state and control to obtain the Jacobian matrices. *ApplyControl* applies the control action to the plant by sending commands to the actuators.

3.3 Baseline controllers

In order to assess the performance of the MPC algorithms above, two baseline controllers are considered for comparison. These are:

1. **A time-varying Linear Quadratic Regulator (LQR):** LQR computes the solution to the Algebraic Ricatti Equation for the linearized model of the system (11). The nonlinear AUV model is linearized periodically and the optimal feedback control from LQR is applied to drive the system to the setpoint. (This is a simpler version of Algorithm 2.)

2. **PID controllers:** Existing trim and flight controllers on SAM are based on PID. Linear PID controllers are used for static depth and pitch control using the trim actuators. Cascaded PID controllers enable control of velocity, heading and diving using the propellers and thrust vectoring.

4 Results

This section is organized as follows. First, the underactuated system's reachability is analysed. Second, results of hydrobatic maneuvers of increasing complexity are presented using the two MPC algorithms in Matlab. Third, the Nonlinear MPC in Algorithm 1 is then validated in the Simulink hydrobatics simulator. Finally, the real-time LTV-MPC from Algorithm 2 is applied on the real hardware, and validated in simulation and experiment. Results from the Stonefish simulator and experiments at a testing facility are presented. The performance of the nonlinear and LTV-MPC are compared to LQR and PID.

4.1 Validation of method

4.1.1 System reachability analysis

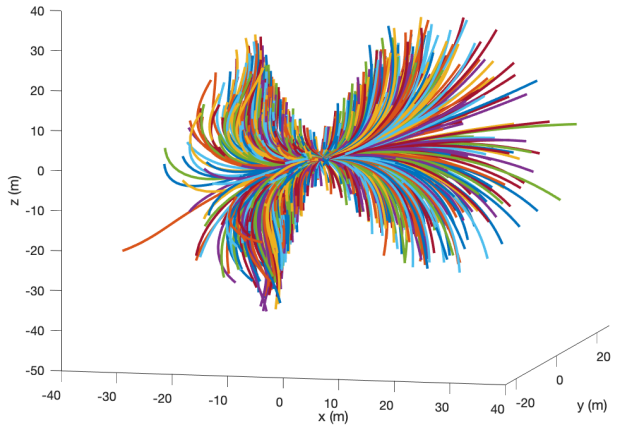
In order to gain a clearer understanding of the behavior of the underactuated system as well as observe potential trajectory deviations due to uncertainty, a reachability validation is performed. Several sets of randomized control inputs within the whole operating range of the different actuators are applied to the system starting from a specific initial condition. The system is time propogated for 20 seconds with these initial control inputs, and the trajectories and terminal states are recorded in Fig. 4a and 4b respectively. It can be seen that using constant controls, uncertainties can lead to a cone of reachable trajectories (Fig. 4a). The point cloud of terminal states (Fig. 4b) appears like a double-funnel with the initial state at the centre due to the limits in the thrust vectoring and trim systems. It can be seen from the extremeties of the funnels that the thrusters are the dominant actuators, but the use of trim subsystems can enable enhanced maneuverability and reachability, especially in the vertical plane. Specific combinations and control sequences are however necessary to cover unreachable states in the point cloud (e.g. for tight turns or vertical hovering), and these cannot be reached by constant controls.

The actuator configuration enables effective control in x, z and θ, while control along the y, ϕ and ψ degrees of freedom is more challenging. Furthermore, if actuation energy is considered, it also costs more to influence the second set (y, ϕ and ψ). When rates are considered, the influence of the propellers and thrust vectoring system make it easy to control the x-velocity and the yaw and pitch rates, but influencing the roll rates, y- and z-velocities is more complicated.

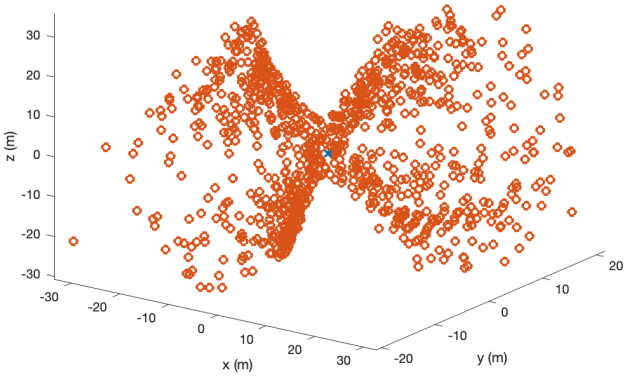
These considerations mean that the use of model based methods can be beneficial in exploiting the dynamics and stability characteristics.

4.1.2 Hydrobatics with Nonlinear MPC

Nonlinear MPC is used to find the optimal control for maneuvering by solving the problem formulated in (6) using SQP through Algorithm 1. In this case, the nonlinear prediction model in equation (5) together with the actuator models is used as the plant to which control sequences are applied. Control strategies for maneuvers of increasing complexity (and requiring the use



(a) Reachable trajectories



(b) Terminal states

Fig. 4: Reachable trajectories (a), terminal states (b) with randomized control inputs starting from $(x, y, z) = (0, 0, 0)$ for 20 seconds.

of different actuator combinations) are studied in order to validate the method used. In each maneuver, the objective function J from (6) was set with a reference output state $\boldsymbol{s}_{\text{ref}}$, output state weight $\boldsymbol{Q} \in \mathbb{R}^{12\times12}$, control weight matrix $\boldsymbol{R_1} \in \mathbb{R}^{6\times6}$ and control rate weight $\boldsymbol{R_2} \in \mathbb{R}^{6\times6}$. The weight matrices were respectively multiplied by the the state vector $\boldsymbol{s}_{\text{ref,out}} = [\ x\ \ y\ \ z\ \ \phi\ \ \theta\ \ \psi\ \ u\ \ v\ \ w\ \ p\ \ q\ \ r\ ]$ and control input vector $\boldsymbol{c} = [\ \text{rpm}_1\ \ \text{rpm}_2\ \ \delta_e\ \ \delta_r\ \ \text{vbs}\ \ \text{lcg}\ ]$ according to (6). In each case, the system was initialized at $\boldsymbol{s}_{\text{out}} = [\ 0\ \ 0\ \ 4\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]$, with all values in SI units. The presented maneuvers are relevant for practical scenarios including transit, inspection, docking and obstacle avoidance. These maneuvers include:

Transit in confined spaces (Fig. 5) Transit is a common scenario in most underwater operations, where the AUV has to go to a desired position some distance away. In transiting to the goal position, the AUV uses the propeller and thrust vectoring systems, with some trim adjustments. The propeller's *rpm* in particular go to the maximum positive and negative values, and the *rpm* control has a bang-bang characteristic.

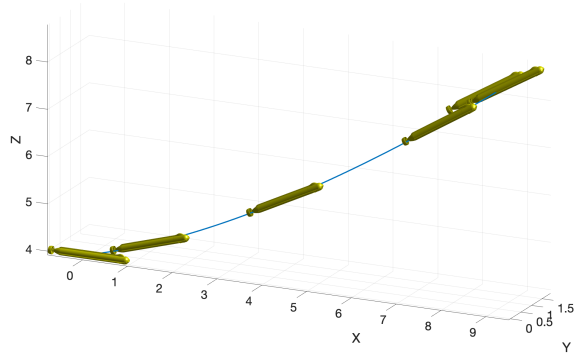
Considering the problem formulation in (6), the objective function $\boldsymbol{J(s, c)}$ for this maneuver includes the following weight matrix $\boldsymbol{Q}$ for deviation from the reference position $\boldsymbol{s}_{\text{ref}}$ and small weights $\boldsymbol{R_1, R_2}$ on the control inputs. Orientations and velocities are unconstrained.

$$\begin{aligned}
\boldsymbol{s}_{\text{ref}} &= [\ 8\ \ 2\ \ 8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ] \\
\boldsymbol{Q} &= \text{diag}([\ 100\ \ 100\ \ 100\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]) \\
\boldsymbol{R_1} &= \text{diag}([\ 10\ \ 10\ \ 10\ \ 10\ \ 1\ \ 1\ ]) \\
\boldsymbol{R_2} &= \text{diag}([\ 10\ \ 10\ \ 10\ \ 10\ \ 1\ \ 1\ ]).
\end{aligned} \tag{12}$$

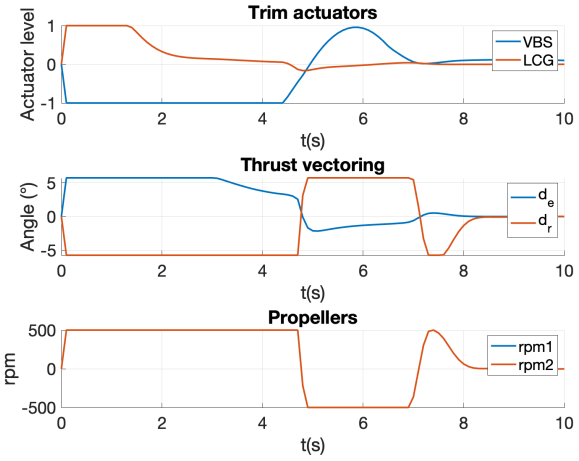
Tight turning for launch, recovery and obstacle avoidance (Fig. 6) During launch, recovery and obstacle avoidance, tight turns and on-the-spot maneuvers can be needed for precise positioning. This becomes a challenge with an underactuated system, since dynamic constraints mean that all states are not easily reachable. However, the MPC suggests a control where the propulsion and thrust vectoring are asynchronously cycled between maximum positive and negative values, enabling an elegant 'turbo-turn'. This asynchronous bang-bang thrust vectoring sequence enables turning on the spot, akin to parallel parking a car.

The objective function for tight turning includes weights $\boldsymbol{Q}$ on deviation from the reference position and orientation, as well as small weights on $\boldsymbol{R_1, R_2}$ the control inputs. For readability, the orientations are presented in degrees but calculations are performed after conversion to radians.

$$\begin{aligned}
\boldsymbol{s}_{\text{ref}} &= [\ 1.5\ \ 3\ \ 4\ \ 0\ \ 0\ \ \text{deg2rad}(60°)\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ] \\
\boldsymbol{Q} &= \text{diag}([\ 100\ \ 100\ \ 100\ \ 0\ \ 0\ \ 100\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]) \\
\boldsymbol{R_1} &= \text{diag}([\ 10\ \ 10\ \ 10\ \ 10\ \ 1\ \ 1\ ]) \\
\boldsymbol{R_2} &= \text{diag}([\ 10\ \ 10\ \ 10\ \ 10\ \ 1\ \ 1\ ]).
\end{aligned} \tag{13}$$

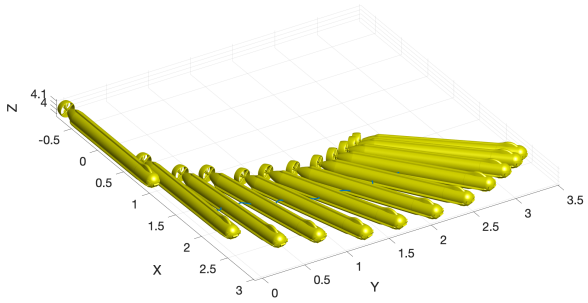


(a) AUV Trajectory

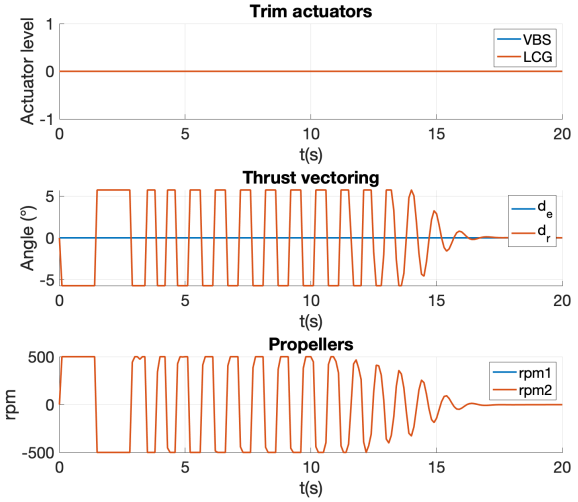


(b) Actuators

Fig. 5: Going to a position (8,2,8) using nonlinear MPC with Algorithm 1 and the Matlab SQP solver. A bang-bang control is seen with the propellers while thrust vectoring, LCG and VBS actuators are used for adjustments. The AUV starts from (0,0,4).

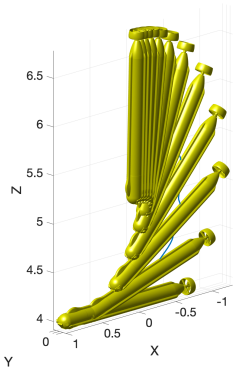


(a) AUV Trajectory

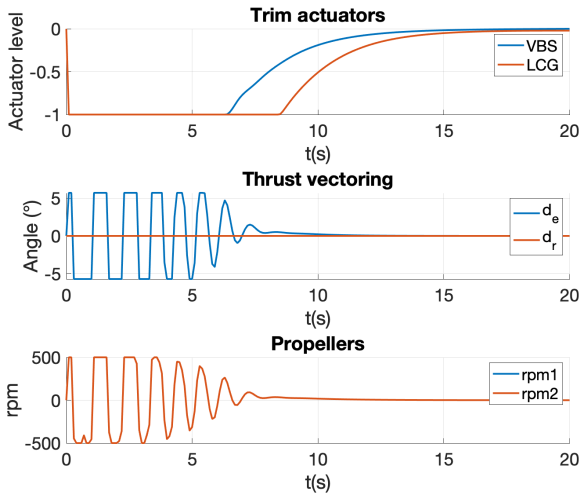


(b) Actuators

Fig. 6: A tight 90 degree turbo-turn using nonlinear MPC with Algorithm 1 and the Matlab SQP solver. The propellers and thrust vectoring are cycled asynchronously between maximum and minimum values.The AUV starts from (0,0,4).

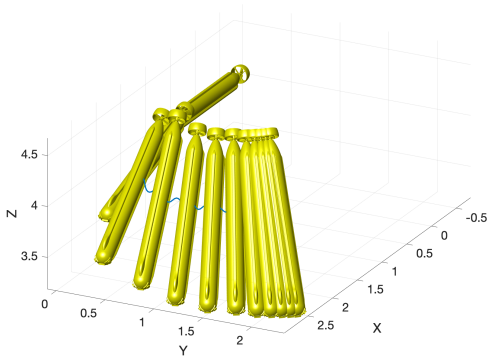


(a) AUV Trajectory

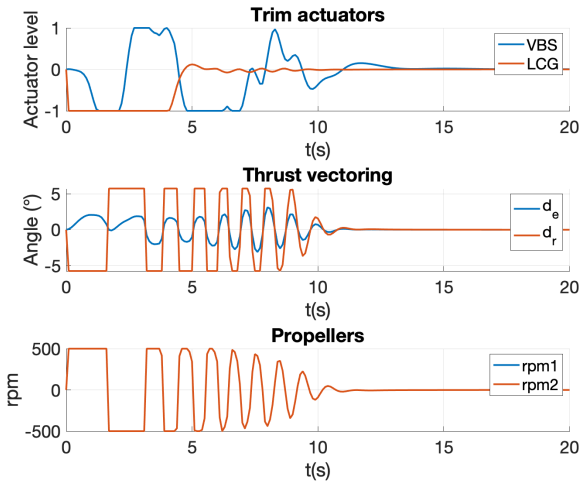


(b) Actuators

Fig. 7: Transition to vertical hovering using nonlinear MPC with Algorithm 1 and the Matlab SQP solver. The propellers and thrust vectoring subsystem are used to begin the maneuver, while the trim subsystems are used to hold the orientation. The AUV starts from (0,0,4).



(a) AUV Trajectory



(b) Actuators

Fig. 8: A helical inspection using nonlinear MPC with Algorithm 1 and the Matlab SQP solver. All actuators are used to different extents to realize this maneuver. The AUV starts from (0,0,4).

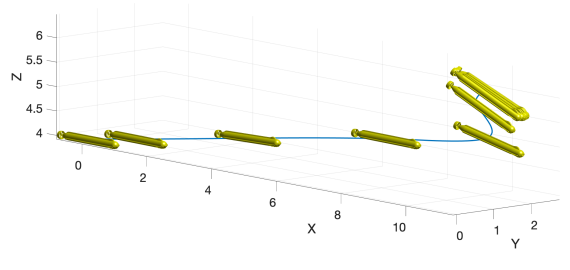
Vertical hovering and agile trimming for inspection (Fig. 7) The trim subsystems can enable static positioning, and allow the AUV to pitch up to 90°. In this maneuver, the controller uses the propulsion and thrust vectoring subsystems to turn the AUV upwards using the turbo-turn sequence, while the trim actuators allow the AUV to stay vertical. Such positioning can, for example, be useful for hovering and docking. To achieve trimming, a high penalty is added in $\boldsymbol{Q}$ to the pitch deviation and the propeller inputs. This encourages the use of the trim subsystems to minimise the pitch error.

$$\begin{aligned}\boldsymbol{s}_{\text{ref}} &= \begin{bmatrix} 0 & 0 & 6 & 0 & 0 & \text{deg2rad}(90^\circ) & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{Q} &= \text{diag}(\begin{bmatrix} 100 & 100 & 100 & 0 & 1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}) \\ \boldsymbol{R_1} &= \text{diag}(\begin{bmatrix} 1000 & 1000 & 100 & 100 & 1 & 1 \end{bmatrix}) \\ \boldsymbol{R_2} &= \text{diag}(\begin{bmatrix} 1000 & 1000 & 100 & 100 & 1 & 1 \end{bmatrix}).\end{aligned} \tag{14}$$

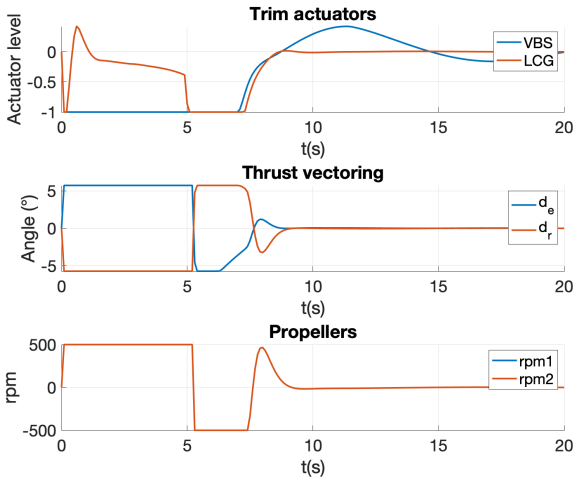
Helical inspection of a confined space (Fig. 8) A combination of the turbo-turn sequence and the trim actuators can also be used for a helical maneuver, where the AUV can perform panoramic inspections of confined spaces, or perform coverage of irregular shapes. Here it can be seen that all actuators are used in a bang-bang sequence. To enable this maneuver, the objective penalises deviation from a reference position and orientation, with minor weights on the control inputs. A reference position close to the start position encourages tighter turning and trimming.

$$\begin{aligned}\boldsymbol{s}_{\text{ref}} &= \begin{bmatrix} 2 & 2 & 4 & 0 & \text{deg2rad}(45^\circ) & \text{deg2rad}(90^\circ) & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{Q} &= \text{diag}(\begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}) \\ \boldsymbol{R_1} &= \text{diag}(\begin{bmatrix} 10 & 10 & 10 & 10 & 1 & 1 \end{bmatrix}) \\ \boldsymbol{R_2} &= \text{diag}(\begin{bmatrix} 10 & 10 & 1 & 1 & 1 & 1 \end{bmatrix}).\end{aligned} \tag{15}$$

Generalized trajectory tracking (Fig. 9) Combining such approaches, the AUV can be made to go to an arbitrary pose in 3 dimensional space, including transit as well as tight maneuvering. In this example, the AUV is commanded to a waypoint, and asked to maintain a 30° pitch angle. This makes it necessary to combine transit and agile maneuvering. Such trajectory tracking can be

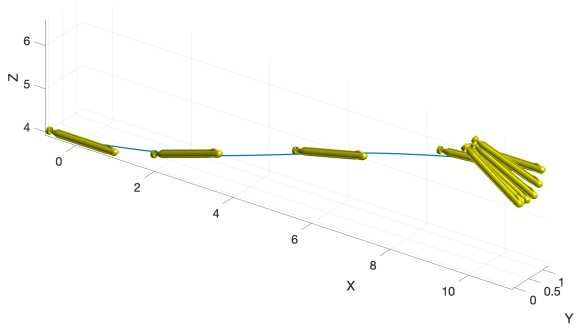


(a) AUV Trajectory

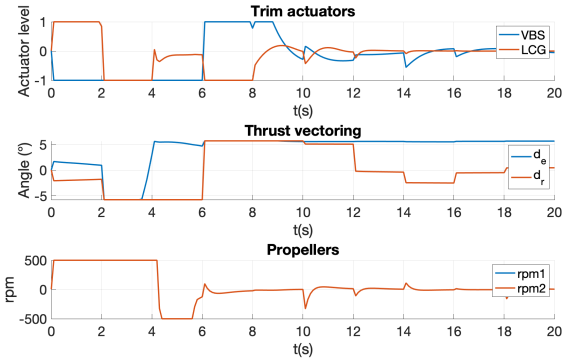


(b) Actuators

Fig. 9: Tracking a general 3D trajectory using nonlinear MPC with Algorithm 1 and the Matlab SQP solver. A bang-bang control sequence is seen, with the propellers used for transit and the trim actuators for final orientation adjustments. The AUV starts from (0,0,4).



(a) AUV Trajectory



(b) Actuators

Fig. 10: Tracking a general 3D trajectory using realtime MPC with Algorithm 2 and the QP solver. The LTV solution appears to be a coarser form of the nonlinear solution. The AUV starts from (0,0,4).

Controller	Solver/Realtime ratio	Tracking Error(m)	Cost
NMPC	2.3	0.17	2.26e6
LTV-MPC	0.2	0.56	2.39e06

Tab. 1: MPC performance comparison for going to a general waypoint in Figures 9 and 10.

achieved with the same objective as the helix maneuver, but with a different reference state.

From these different maneuvers, it can be seen that the optimal control balances the use of the available actuators, uses the propellers for transit, and the LCG and VBS for fine adjustments and tight maneuvers. The optimal control using NMPC tends towards bang-bang control in the absence of disturbances.

4.1.3 Linearization for real-time control

The nonlinear controller is linearized periodically for real-time control using Algorithm 2; and the results of the real-time controller for generalized trajectory tracking are presented in Fig. 10. The time-varying linear MPC (LTV-MPC) shows qualitatively similar controls and behaviors as the nonlinear MPC, but the linearization leads to some deviations from the NMPC solution.

Qualitatively, the performance of the real-time controller is a coarser solution of the NMPC, with a finer linearization resolution leading to a control that is closer to the nonlinear solution. Three linearizations are considered, and the deviation is presented in Fig 11. It can be seen that linearizing the model with a fine resolution of every 20 time steps offers a closer approximation of the nonlinear controller when compared to a coarse resolution of 100 time steps. However, if the model is linearized too frequently (< 5 timesteps), the solver sometimes fails to converge due to insufficient iterations during that interval. A resolution > 10 timesteps has been used in this paper to address this issue.

Table 1 presents a comparison between the tracking performance, cost (a measure of efficiency) and solver time ratio (where a value ≥ 1.0 means the control is slower than real-time) of the two approaches. It can be seen that the LTV-MPC with a resolution of 20 time steps has a significant advantage in solver time, but with a trade-off in cost and tracking.

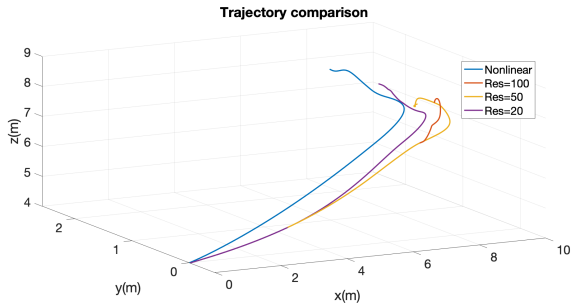


Fig. 11: Deviations from NMPC based on linearization resolution. the parameter *Res* indicates the number of time-steps between successive linearizations.

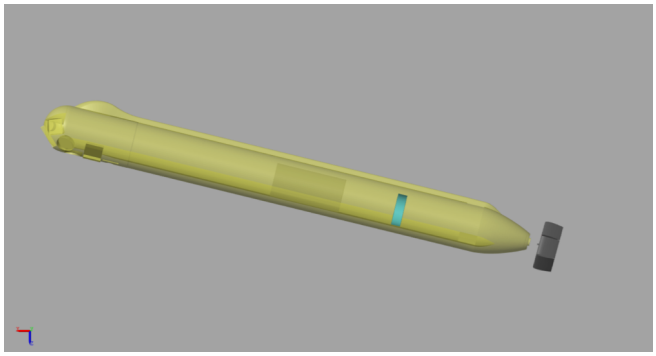


Fig. 12: The AUV model in the Simulink hydrobatics simulator

4.2 Implementation in simulation and hardware

4.2.1 Nonlinear MPC implementation in Simulink

The tracking and robustness performance of the nonlinear MPC is evaluated in the SMaRC hydrobatics simulator in Simulink (detailed technical information on the simulator can be found in [Bhat et al., 2021]). The simulator uses a full-envelope hydrodynamic model for an AUV and exhibits dynamic behavior similar to the real vehicle even at high angles of attack. The AUV model is integrated with ROS and it is possible to auto-generate C-code in Simulink for hardware deployment. Evaluation on this simulator is a step closer to running the MPC on the robot hardware.

The nonlinear MPC's tracking performance is tested on two maneuvers - straight line transit and vertical ascent, and the performance is compared to a time varying Linear Quadratic Regulator (LQR) and a PID controller. Furthermore, to test robustness to disturbances, random noise (having mean amplitude of 0.8, variance of 0.25, sample time of 0.1s leading to a signal-to-noise ratio of 20dB) is added to the state feedback. The results are presented in Figures 13, 14 and 15. It can be seen that the NMPC has significantly better robustness to disturbances in comparison to the LQR and PID, with reasonable performance in terms of rise time, overshoot and steady state error. More information on the LQR implementation can be found in [Panteli, 2019].

Detailed results of the controller performance com-

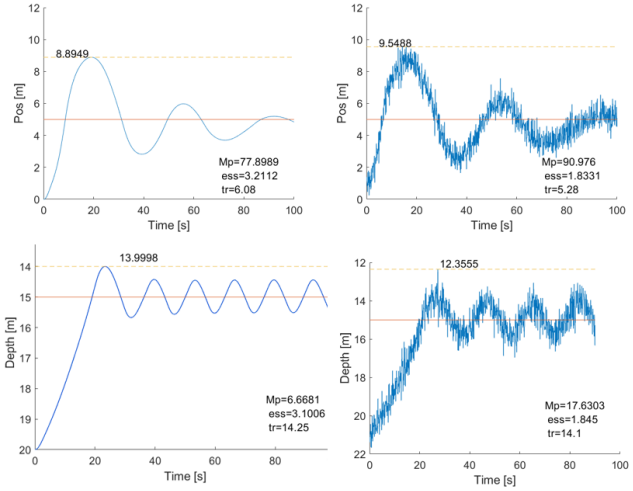


Fig. 13: Nonlinear MPC in Simulink, Top left: Horizontal trajectory without noise, Bottom left: Vertical trajectory without noise, Top right: Horizontal trajectory with noise, Bottom right: Vertical trajectory with noise.

parison are presented in Table 2. The nonlinear MPC exhibits a significantly lower steady state error than the PID and LQR for most cases. The NMPC steady state error was as low as 10% of the PID error and 18% of the LQR error in the maneuvers. The MPC has a slower rise time (up to 2x the rise time of LQR) in the horizontal case where the propellers are used, but a faster rise time (up to 40% the rise time of PID) in the vertical case, where the buoyancy system is used. NMPC has a higher overshoot than LQR (up to 2x the LQR overshoot), and a comparable overshoot to PID. However, as expected the NMPC uses more processing power than the other two controllers, as is seen from a higher run time/simulation time ratio in general (a value closer to 1 signifies real-time performance). It can be seen that for simpler maneuvers such as these, the LQR and PID have competitive performance, though when maneuver complexity and robustness requirements increase (such as in the turbo-turn or trim control maneuvers), the LQR and PID cannot always be used and the NMPC offers better performance. Furthermore, in the presence of disturbances, the LQR has a significantly increased rise time (up to 2x, see Fig. 14) compared to a disturbance-free case, while the PID sometimes fails completely (e.g in a horizontal trajectory, see Fig. 15). The NMPC's performance however, is relatively unaffected at the same noise level .

4.2.2 LTV-MPC experiments on the SAM AUV

For testing in real-time on the robot hardware, the LTV-MPC based on Algorithm 2 has been implemented in Python and ROS using the CVXPY optimization library and the OSQP solver. The experimental scenario focuses on simultaneous pitch and depth control of SAM using the trim (LCG) and buoyancy (VBS) actuators within

Maneuver	Controller	Rise time(s)	Overshoot(%)	S.S.error(%)	Solver/Realtime ratio
Straight line transit	NMPC	5.60	50.0	2.22	6.42
	LQR	2.25	24.5	12.56	5.90
	PID	4.56	64.66	20.9	7.13
Vertical ascent	NMPC	12.71	13.29	2.23	8.23
	LQR	19.07	8.006	3.84	6.24
	PID	29.08	10.91	14.849	5.73

Tab. 2: NMPC performance chart, horizontal transit and vertical ascent trajectories in the Simulink Hydrobatics Simulator. All results presented are averages of 3 repeated runs.

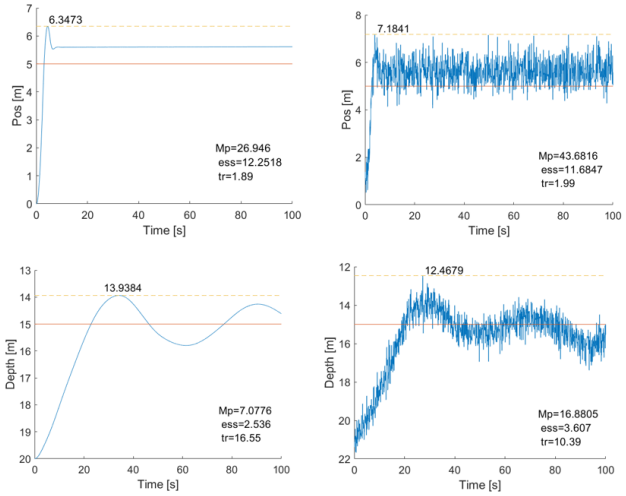


Fig. 14: LQR in Simulink, Top left: Horizontal trajectory without noise, Bottom left: Vertical trajectory without noise, Top right: Horizontal trajectory with noise, Bottom right: Vertical trajectory with noise.

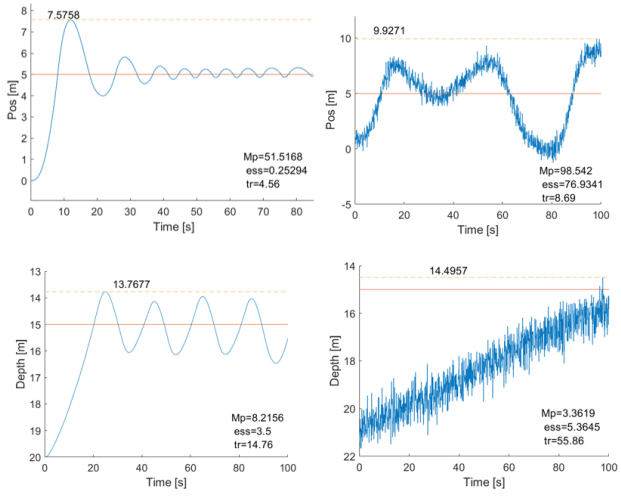


Fig. 15: PID in Simulink, Top left: Horizontal trajectory without noise, Bottom left: Vertical trajectory without noise, Top right: Horizontal trajectory with noise, Bottom right: Vertical trajectory with noise.

a confined environment. The weights used are the similar to the simulated trimming maneuver in (14). This testing scenario is relevant for inspection, manipulation and hovering in confined environments, and requires agile maneuvering. It places high requirements on control due to close coupling between the pitch and depth states. Another factor in this scenario is limited sensor feedback to the EKF used for state estimation. The nature of the confined environment means we have good orientation and depth feedback from the compass, IMU and pressure sensors, but bad positioning from the DVL due to reflections.

In order to verify the control algorithms and check system integration, the test scenario is first rehearsed in the Stonefish Simulator, comparing the LTV-MPC to time-varying LQR and PID controllers.

The results of the MPC, LQR and PID to hold a depth of 1.5m and pitch angle of 6 degrees in the Stonefish simulator are presented in Figures 17, 18 and 19. An error threshold $\epsilon = 0.1m$ in depth was set for the MPC and LQR. Looking at detailed results in Table 3, it can be seen that the MPC has a lower rise time than the LQR and PID ($< 50\%$ of PID). MPC has similar overshoot behavior as LQR and PID (with a relative difference of 20%), while MPC and PID have significantly lower steady state error in comparison to LQR. It can be seen that the MPC has the best overall performance overshoot, steady-state error, rise time and settling time than the LQR and PID. The presented cost is the summation of the objective function J in (6), including the state error and control usage over the full period of control. A lower cost would imply lower error and actuator usage. From this quantitative cost summation, MPC has a lower cost than PID and LQR. Looking at the result plots, it can also be seen qualitatively that the actuator usage, error and oscillations are minimum with MPC.

Following the rehearsal in Stonefish, the experimental scenario was tested with the SAM AUV in an indoor test tank at Saab Dynamics, Linköping, Sweden (see Fig. 20). The tank is 6m deep and 10m in diameter, and offers a controlled environment for repeatable experiments. The MPC and PID controllers were used to control SAM and the controller performance was compared. The MPC was run at a frequency of 10hz (meaning a timestep of 0.1s) and linearized every 20 timesteps, while the PID was run at 50hz. Note that, due to hardware failure the PID controllers could not be run with the exact same setpoints as the MPC, and the LQR controllers could not be tested.

The experimental results of the LTV-MPC on SAM are presented in Fig. 21. It can be seen that the MPC

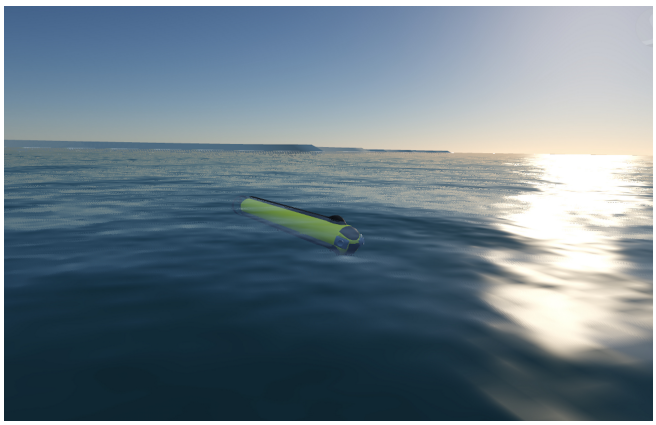


Fig. 16: SAM in the Stonefish simulator. A simulation video can be viewed in Video 1 in the supplementary material.

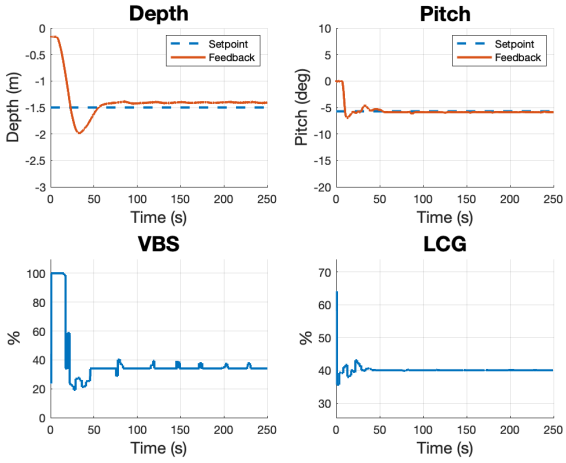


Fig. 17: LTV-MPC in Stonefish

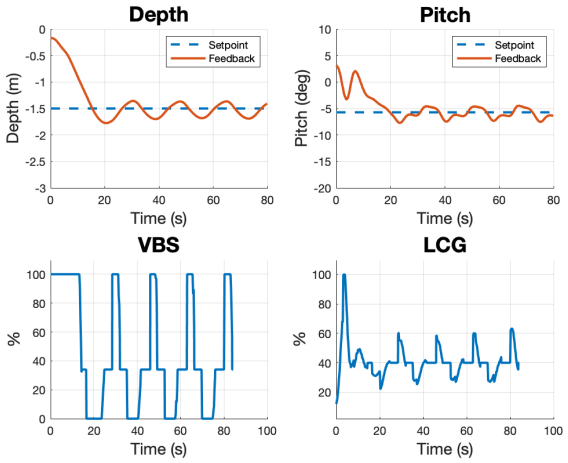


Fig. 18: Time-varying LQR in Stonefish

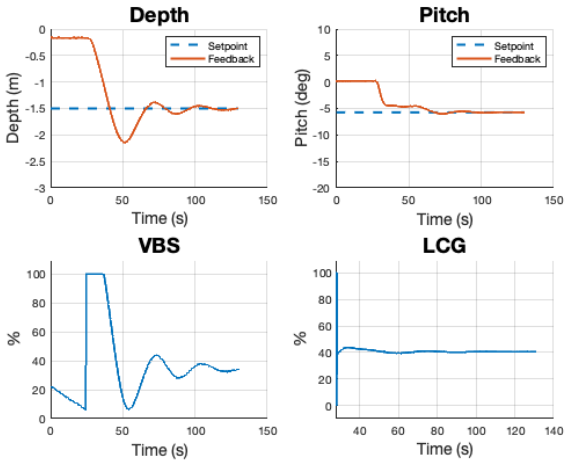


Fig. 19: PID in Stonefish

holds the depth at 1.5m with a deviation of up to 0.2m with few oscillations. The pitch angle is maintained at -6 degrees within a precision of ±1 degrees. In contrast, the PID controller has higher steady state errors (47% higher than MPC) and longer settling time due to the coupling between the states. It is also interesting to observe that the actuator usage is significantly less in the case of MPC compared to PID - the LCG actuator is heavily used to compensate for center of gravity and center of buoyancy changes with the PID, while the MPC predicts these changes and pre-empts them with the actuator commands. Quantitatively as well, on comparing the summation of the objective function, the MPC has a 7% lower cost.

Detailed results in terms of overshoot, rise-time, settling time and steady-state error of the different evaluated controllers in Stonefish and in the experiment are presented in Table 3. It is clear that the LTV-MPC offers good performance at low cost. This means that it is possible to achieve real-time control on target hardware with the presented model predictive control approach.

Datasets and videos from the Stonefish simulations and tank experiments are available in the supplementary material.

5 Discussion

5.1 Numerical simulations with nonlinear MPC

Simulating maneuvers of increasing complexity, following a stability and reachability validation, enables a sanity check of the nonlinear MPC method. Simpler maneuvers such as transiting to a waypoint, or performing a turbo-turn have intuitive controls, and the computed optimal solution aligns with intuition. The existence of the bang-bang results motivates the optimality of the solution. When more complex maneuvers such as vertical hovering or helices are evaluated, actuators are exploited based on their relative strengths (including the use of bang-bang sequences). Such control solutions considering the dynamics model and constraints cannot be easily obtained by linear feedback control methods. The sequential quadratic programming solver converged in all these cases, meaning the 'optimum' cost has been found.

The solution time is the main limitation of the nonlinear MPC implementation (this has also been noted in [Shen et al., 2016, Steenson et al., 2014]). A Solver/Realtime ratio ≤ 1 is beneficial for real-time con-

Context	Controller	Mean Rise Time(s)	Mean Overshoot(%)	Mean S.S.error(%)	Cost
Stonefish Simulator	LTV-MPC	21.0	24.17	3.42	5.80e7
	LQR	21.6	24.5	22.5	5.81e7
	PID	54.15	19.17	0.42	5.82e7
Experiment	LTV-MPC	45.83	33.67.0	15.0	8.39e7
	PID	34.0	39.77	23.18	9.08e7

Tab. 3: Results from the experimental scenario considering static pitch and depth control of the SAM AUV in a confined environment.

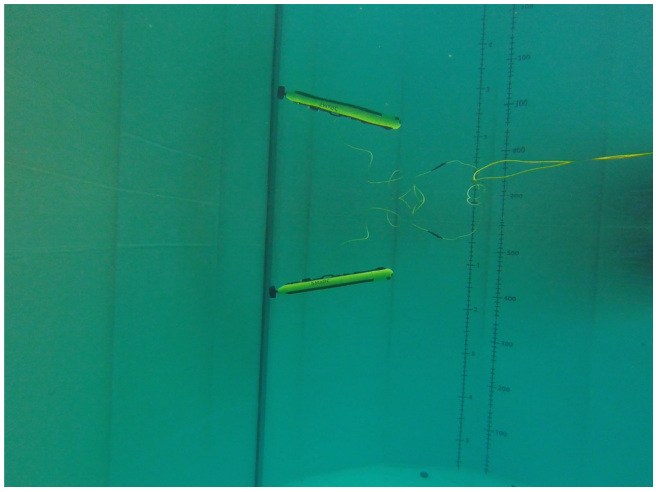


Fig. 20: The SAM AUV during the MPC tests at Linköping, Sweden. A video from the experiment can be viewed in Video 1 in the supplementary material.

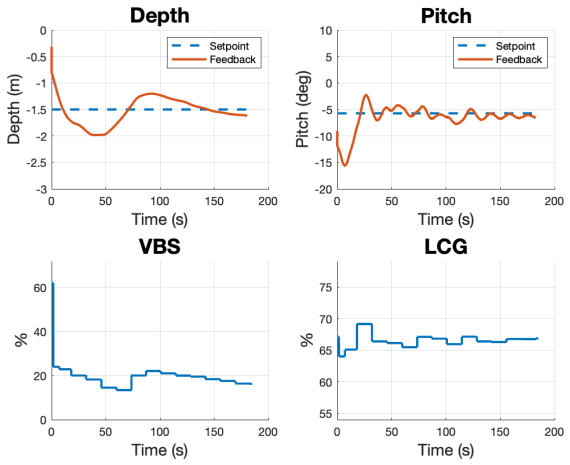


Fig. 21: LTV-MPC: Experimental results on SAM to control pitch and depth

Fig. 22: PID: Experimental results on SAM to control pitch and depth

trol implementations. However, the nonlinear control problem could only be solved slower than real-time, and the implementation is not yet fast enough for online implementations. With further evaluation and code optimization, even the nonlinear MPC may be real-time applicable. Notably though, the optimal solutions of the nonlinear MPC provided an expert demonstration, enabling evaluation of other sub-optimal strategies.

Linearizing the nonlinear MPC periodically according to Algorithm 2 enables faster solutions, but at the cost of deviating from the optimum. This is clear on observing the optimal control solutions in Figs. 9 and 10. The linearized solution appears as an approximation of the optimal solution if the resolution is fine enough, though deviations occur at coarse resolutions (Fig. 11). Overall, such a linearization strategy appears suitable for real-time control applications with a fine enough resolution. Practically, for fast maneuvers, a 'fine' resolution of linearizations every 20 time-steps offers good performance. Finer resolutions (< 5 time-steps) sometimes lead to a lack of solver convergence. For slower maneuvers such as trim control, a coarser resolution of 50 time-steps has been sufficient.

5.2 MPC on robot hardware

Interesting results have been obtained on testing the real-time MPC within the ROS ecosystem both on robot hardware and in simulation. The existing AUV control system uses a set of cascaded PID controllers for depth and heading control, linear PID trim controllers for buoyancy and pitch control, and scripted sequences for advanced maneuvers, all regulated by a behavior tree (with safety checks). Using MPC can reduce the complexity of this setup, while also exploiting the couplings between states and actuators. Maneuver sequences for launch and recovery, as well as precise inspections can benefit significantly from the model predictive controller. MPC can simplify some complex actuator sequences and enable tight maneuvers with a lower 'cost', but with higher computational load. In general, MPC enables elegant use of the available actuators to perform hydrobatic maneuvers, and this has been demonstrated in experiments. Very fast maneuvers have not yet been tested on hardware using MPC- it may be necessary to balance the linearization resolution to effectively reflect the dynamics while offering real-time control.

Two key challenges have been observed during testing. First, a lack of solver convergence may lead to outlier solutions and numerical instability. This issue has been addressed by filtering the MPC output, and adding safety checks with a behavior tree. Second, accurate state feedback is very difficult to obtain underwater due to high sensor uncertainty. This has a direct effect on the control, since the MPC solver may not have an accurate

estimate of the error it has to minimize. This has been partially addressed by using an Extended Kalman Filter for improving state estimation, but accurate underwater localization is still a challenging open problem in the field.

Implementing and evaluating the nonlinear MPC in Simulink with the hydrobatics simulator has been a crucial step towards running nonlinear MPC in real-time on hardware. Solver optimization and automatic C-code generation are logical next steps to deploy the controllers from the ROS environment in Simulink to the actual AUV. Preliminary tests show promise, but comprehensive evaluation is the scope of future work.

Operationally, it makes sense to use the most effective control actions for specific mission segments. It would be more efficient to use a simple PID controller for long transits, while the same would show limitations in tight maneuvers. Similarly, MPC can be effective in challenging scenarios, but might be too expensive computationally for simpler maneuvers. In some critical scenarios such as emergency ascent, it even makes sense to use scripted sequences based on motion primitives instead of feedback control. Therefore, effectively switching between controllers, and selecting a prioritization logic is key to optimizing performance and robustness. The behavior tree automaton in our implementation offers a flexible framework for such switching, and several interesting combinations are yet to be explored.

5.3 Augmentations

Uncertainty estimates are not part of this work – robustness issues are addressed by adding measurement noise, using a full envelope nonlinear model in the hydrobatics simulator (presented in [Bhat et al., 2021]), and extending the MPC to a hybrid automaton using behavior trees [Marzinotto et al., 2014] to account for unsafe conditions (with safety limits and emergency actions [Bhat et al., 2020]). Furthermore, robustness is inherent in MPC because of its feedback nature, so a certain degree of disturbance rejection and robustness to uncertainty exists.

Augmentations can be made to improve the performance of the real-time MPC. Improving robustness in the presence of uncertainty is a key consideration. Including uncertainty models within the optimization problem can improve robustness to uncertainties. Another robustness improvement can be to include feedback motion planning libraries [Majumdar and Tedrake, 2017] or quasi-infinite horizon solutions [Chen and Allgöwer, 1997] with sets of globally stabilising trajectories, so that regions of attraction can be exploited. The use of a nominal control u_0 is a first step in this direction.

The prediction model used is crucial to controller performance, since an inaccurate prediction model will lead to incorrect controls. System identification techniques (e.g. SINDy, Koopman operators, Lagrangian Neural Networks, physics-informed learning) can be used to update the dynamics model from sensor data. This can enable adaptive control while reducing the model uncertainty.

The hydrobatic maneuver sequences derived offline through nonlinear MPC can also be used as motion primitives. These primitives could be composed into hybrid controllers using formal methods and sequential logic, thus generating fast yet robust control strategies.

An alternate method to enable nonlinear MPC in real time is to use the nonlinear MPC solution as an expert demonstrator. Methods in supervised learning and reinforcement learning can then be used to learn the optimal policy from the offline MPC solution. The trained agents can then be deployed on hardware, thus reducing computing time significantly. A limitation here, of course, is the lack of stability guarantees, but this could be addressed by hard safety limits.

6 Conclusion

In this paper, nonlinear model predictive control has been applied to underactuated AUVs. Nonlinear MPC has been used to study hydrobatic maneuvers including transits, turbo-turns, helices, and static hovering in numerical simulations. For real-time applications, the nonlinear MPC has been periodically linearized to obtain a Linear Time Varying MPC. Computationally, the linearized controllers require significantly lower solver time (down to the order of 1/100) compared to the nonlinear solution, while deviations from the optimum stay in a manageable level (up to 20%) for the maneuvers studied.

The nonlinear MPC algorithm has been validated in Simulink, and shows better tracking performance and robustness than baseline LQR and PID controllers. The LTV-MPC has been implemented in ROS using CVXPY and tested on the SAM AUV hardware. Simulations and field experiments with SAM have shown positive results, where the LTV-MPC outperforms PID and LQR in terms of rise time, settling time, overshoot and steady-state error as well as minimizes actuator usage.

The field tests have been one of the first demonstrations of the use of MPC for hydrobatics with AUV hardware. Such field experiments demonstrate the applicability of real-time MPC in AUV missions. While the focus here has been on real-time control, future work will focus on augmentations to improve robustness, stability and adaptability.

Acknowledgement

This work was supported by the Swedish Foundation for Strategic Research (SSF) through the Swedish Maritime Robotics Center (SMaRC). The authors acknowledge Josefine Severholt, Anna Arnwald and Carl Ljung for their support with hardware and field tests. The authors would also like to thank Jan Siesjö at Saab Dynamics for the opportunity to utilize testing facilities belonging to Saab in Linköping.

References

[Abbeel et al., 2007] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1–8. MIT Press.

[Amos et al., 2018] Amos, B., Rodriguez, I. D. J., Sacks, J., Boots, B., and Kolter, J. Z. (2018). Differentiable mpc for end-to-end planning and control. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8299–8310, Red Hook, NY, USA. Curran Associates Inc.

[Bangura and Mahony, 2014] Bangura, M. and Mahony, R. (2014). Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780. 19th IFAC World Congress.

[Berberich et al., 2020] Berberich, J., Koehler, J., Muller, M. A., and Allgower, F. (2020). Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, page 1–1.

[Berg and Wicklander, 2016] Berg, R. and Wicklander, H. (2016). Swedish aip submarine development. In *in Undersea Defence Technology Conference (UDT 2016)*, Oslo, Norway.

[Betts, 2010] Betts, J. T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, second edition.

[Bhat and Stenius, 2018] Bhat, S. and Stenius, I. (2018). Hydrobatics: A review of trends, challenges and opportunities for efficient and agile underactuated auvs. In *2018 IEEE/OES AUV 2018 , Porto*.

[Bhat et al., 2019] Bhat, S., Stenius, I., Bore, N., Severholt, J., Ljung, C., and Torroba Balmori, I. (2019). Towards a cyber-physical system for hydrobatic auvs. In *OCEANS 2019 - Marseille*, pages 1–7.

[Bhat et al., 2021] Bhat, S., Stenius, I., and Miao, T. (2021). Real-time flight simulation of hydrobatic auvs over the full 0°–360° envelope. *IEEE Journal of Oceanic Engineering*, 46(4):1114–1131.

[Bhat et al., 2020] Bhat, S., Torroba, I., Özkahraman, Ö., Bore, N., Sprague, C. I., Xie, Y., Stenius, I., J., J. S., Ljung, C., Folkesson, J., and Ögren, P. (2020). A cyber-physical system for hydrobatic auvs: System integration and field demonstration. In *IEEE OES AUV2020 Symposium*, St. John's, NL, Canada.

[Bosch et al., 2015] Bosch, J., Ridao, P., Ribas, D., and Gracias, N. (2015). Creating 360° underwater virtual tours using an omnidirectional camera integrated in an auv. *OCEANS 2015 - Genova*, pages 1–7.

[Bulka and Nahon, 2018] Bulka, E. and Nahon, M. (2018). Automatic control for aerobatic maneuvering of agile fixed-wing uavs. *Journal of Intelligent & Robotic Systems*.

[Chen and Allgöwer, 1997] Chen, H. and Allgöwer, F. (1997). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. In *1997 European Control Conference (ECC)*, pages 1421–1426.

[Cory and Tedrake,] Cory, R. and Tedrake, R. *Experiments in Fixed-Wing UAV Perching*.

[Duecker et al., 2021] Duecker, D. A., Horst, C., and Kreuzer, E. (2021). From aerobatics to hydrobatics: Agile trajectory planning and tracking for micro underwater robots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8617–8624.

[Fan et al., 2020] Fan, D. D., akbar Agha-mohammadi, A., and Theodorou, E. A. (2020). Deep learning tubes for tube mpc.

[Ferreira et al., 2012] Ferreira, B. M., Jouffroy, J., Matos, A. C., and Cruz, N. A. (2012). Control and guidance of a hovering auv pitching up or down. In *2012 Oceans*, pages 1–7.

[Fossen, 2011] Fossen, T. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd.

[Gomes and Pereira, 2018] Gomes, R. and Pereira, F. L. (2018). A general attainable-set model predictive control scheme. application to auv operations**the authors acknowledge the partial support of fct rd unit systec - poci-01-0145-feder-006933/systec funded by erdf — compete2020 — fct/mec — pt2020 - extension to 2018, project stride - norte-01-0145-feder-000033, funded by erdf — norte 2020, and project magic - poci-01-0145-feder-032485 - funded by feder via compete 2020 - poci, and by fct/mctes via piddac. *IFAC-PapersOnLine*, 51(32):314–319. 17th IFAC Workshop on Control Applications of Optimization CAO 2018.

[Gomes and Pereira, 2019] Gomes, R. and Pereira, F. L. (2019). Model predictive control for autonomous underwater vehicles. *Procedia Computer Science*, 150:19–27. Proceedings of the 13th International Symposium "Intelligent Systems 2018" (INTELS'18), 22-24 October, 2018, St. Petersburg, Russia.

[Gros et al., 2012] Gros, S., Quirynen, R., and Diehl, M. (2012). Aircraft control based on fast non-linear mpc multiple-shooting. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1142–1147.

[Heshmati-alamdari et al., 2018] Heshmati-alamdari, S., Karras, G. C., Marantos, P., and Kyriakopoulos, K. J. (2018). A robust model predictive control approach for autonomous underwater vehicles operating in a constrained workspace.

[Kaufmann et al., 2020] Kaufmann, E., Loquercio, A., Ranftl, R., Müller, M., Koltun, V., and Scaramuzza, D. (2020). Deep drone acrobatics.

[Lorenzen et al., 2019] Lorenzen, M., Cannon, M., and Allgöwer, F. (2019). Robust mpc with recursive model update. *Automatica*, 103:461–471.

[Lu and Cannon, 2019] Lu, X. and Cannon, M. (2019). Robust adaptive tube model predictive control. In *2019 American Control Conference (ACC)*, pages 3695–3701.

[Majumdar and Tedrake, 2017] Majumdar, A. and Tedrake, R. (2017). Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982.

[Marzinotto et al., 2014] Marzinotto, A., Colledanchise, M., Smith, C., and Ögren, P. (2014). Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427.

[Mayne et al., 2000] Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

[Mayne et al., 2011] Mayne, D. Q., Kerrigan, E. C., van Wyk, E. J., and Falugi, P. (2011). Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353.

[Moore and Tedrake, 2012] Moore, J. and Tedrake, R. (2012). Control synthesis and verification for a perching uav using lqr-trees. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3707–3714.

[Mueller and D'Andrea, 2013] Mueller, M. W. and D'Andrea, R. (2013). A model predictive controller for quadrocopter state interception. In *2013 European Control Conference (ECC)*, pages 1383–1389.

[Nielsen et al., 2018] Nielsen, M. C., Johansen, T. A., and Blanke, M. (2018). Cooperative rendezvous and docking for underwater robots using model predictive control and dual decomposition. In *2018 European Control Conference (ECC)*, pages 14–19.

[Nikou et al., 2018] Nikou, A., Verginis, C. K., and Dimarogonas, D. V. (2018). A tube-based mpc scheme for interaction control of underwater vehicle manipulator systems. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6.

[Nikou et al., 2021] Nikou, A., Verginis, C. K., Heshmati-alamdari, S., and Dimarogonas, D. V. (2021). A robust non-linear mpc framework for control of underwater vehicle manipulator systems under high-level tasks. *IET Control Theory & Applications*, 15(3):323–337.

[Özkahraman and Ögren, 2020] Özkahraman, Ö. and Ögren, P. (2020). Underwater caging and capture for autonomous underwater vehicles. In *Global OCEANS*. IEEE.

[Panteli, 2019] Panteli, C. (2019). Comparison of control strategies for manipulating a hydrobatic autonomous underwater vehicle. Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden.

[Ru and Subbarao, 2017] Ru, P. and Subbarao, K. (2017). Nonlinear model predictive control for unmanned aerial vehicles. *Aerospace*, 4(2).

[Saback et al., 2020] Saback, R. M., Conceicao, A. G. S., Santos, T. L. M., Albiez, J., and Reis, M. (2020). Nonlinear model predictive control applied to an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 45(3):799–812.

[Shen et al., 2015] Shen, C., Shi, Y., and Buckham, B. (2015). Model predictive control for an auv with dynamic path planning. In *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 475–480.

[Shen et al., 2016] Shen, C., Shi, Y., and Buckham, B. (2016). Nonlinear model predictive control for trajectory tracking of an auv: A distributed implementation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5998–6003.

[Shen et al., 2019] Shen, C., Shi, Y., and Buckham, B. (2019). Path-following control of an auv: A multiobjective model predictive control approach. *IEEE Transactions on Control Systems Technology*, 27(3):1334–1342.

[Silva and Sousa, 2008] Silva, J. and Sousa, J. (2008). Models for simulation and control of underwater vehicles. In Aschemann, H., editor, *New Approaches in Automation and Robotics*, chapter 11. IntechOpen, Rijeka.

[Steenson et al., 2014] Steenson, L. V., Turnock, S. R., Phillips, A. B., Harris, C., Furlong, M. E., Rogers, E., Wang, L., Bodles, K., and Evans, D. W. (2014). Model predictive control of a hybrid autonomous underwater vehicle with experimental verification. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 228(2):166–179.

[Tedrake et al., 2010] Tedrake, R., Manchester, I. R., Tobenkin, M., and Roberts, J. W. (2010). Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052.

[Wilson, 2009] Wilson, P. A. (2009). Autonomous homing and docking tasks for an underwater vehicle. *IFAC Proceedings Volumes*, 42(18):304 – 309. 8th IFAC Conference on Manoeuvring and Control of Marine Craft.

[Yan et al., 2020] Yan, Z., Gong, P., Zhang, W., and Wu, W. (2020). Model predictive control of autonomous underwater vehicles for trajectory tracking with external disturbances. *Ocean Engineering*, 217:107884.

[Zhang et al., 2016] Zhang, T., Kahn, G., Levine, S., and Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search.