2017

IDVD-based trajectory generator for autonomous underwater docking operations

Yazdani, A.M.; Sammut, K.; Yakimenko, O.A.; Lammas, A.; Tang, Y.; Zadeh, S.M.

Yazdani, A.M., Sammut, K., Yakimenko, O.A., Lammas, A., Tang, Y. and Zadeh, S.M., 2017. IDVD-based trajectory generator for autonomous underwater docking operations. Robotics and Autonomous Systems, 92, pp.12-29.
https://hdl.handle.net/10945/58171

Robotics and Autonomous Systems 92 (2017) 12–29

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

IDVD-based trajectory generator for autonomous underwater docking operations

A.M. Yazdani [a,*], K. Sammut [a], O.A. Yakimenko [b], A. Lammas [a], Y. Tang [a], S. Mahmoud Zadeh [a]

[a] Centre for Maritime Engineering, Control and Imaging, School of Computer Science, Engineering and Mathematics, Flinders University, Adelaide, SA 5042, Australia

[b] Department of Systems Engineering / Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, 93943-5194, United States

HIGHLIGHTS

- A novel IDVD-based guidance system is presented for underwater docking operations.
- The IDVD method performance is examined for offline and online docking scenarios.
- Results indicate the efficiency and applicability of the IDVD-based guidance system.

ARTICLE INFO

Article history:
Received 16 August 2016
Received in revised form 15 December 2016
Accepted 1 February 2017
Available online 27 February 2017

Keywords:
AUV
Underwater docking
Trajectory optimization
IDVD method
Legendre–Gauss–Lobatto pseudospectral method

ABSTRACT

This paper investigates capability and efficiency of utilizing the inverse dynamics in the virtual domain (IDVD) method to provide the real-time updates of feasible trajectory for an autonomous underwater vehicle (AUV) during underwater docking operations. The applicability of the IDVD method is examined for two scenarios. For the first scenario, referred to as an offline scenario, a nominal trajectory may be generated ahead of time based on a priori knowledge about the docking station (DS) pose (position and orientation). The second scenario, referred to as an online scenario, assumes some uncertainty in the DS pose; hence, the reference trajectory needs to be constantly recomputed in real time based on the updates about the DS pose. The offline scenario solution serves as a benchmark solution to check feasibility and optimality of generated trajectory subject to constraints on the states and controls. In particular, the offline solution can assist in making informed trade-off decisions between optimality of solution and computational efficiency. For the relatively simple offline scenario, the IDVD-method solution is compared with the Legendre–Gauss–Lobatto pseudo-spectral (LGLPS) method solution. The software-in-the-loop simulations and Monte Carlo trials are run for robustness assessment. Finally, the potential for the IDVD method to work online, in a closed-loop guidance system, is explored using a realistic cluttered operational simulation environment. Simulation results show that the IDVD-method based guidance system guarantees a reliable and efficient docking process by generating computationally efficient, feasible and ready to be tracked trajectories.

1. Introduction

Autonomous underwater vehicles (AUVs) are increasingly becoming more widely used as platforms for exploring the underwater environment. They are capable of conducting a variety of tasks, including mapping, surveillance, surveying, object localization, and sampling, to name but a few. Although AUVs are in principle capable of operating for long periods over large areas, they are in reality limited by the energy storage capacity of the battery technology that they employ. Finding a means of extending the operational duration of the vehicle especially for long term missions, is therefore vital. The availability of suitably located docking stations (DS), on the other hand, provides an opportunity for the AUV to periodically recharge its battery and transfer data while minimizing the time and energy otherwise wasted on the AUV recovery by a mothership whenever the AUV's batteries are depleted. In order to drive an AUV into a DS, the AUV must be equipped with a guidance system that is capable

* Corresponding author.
E-mail address: amirmehdi.yazdani@flinders.edu.au (A.M. Yazdani).

http://dx.doi.org/10.1016/j.robot.2017.02.001
0921-8890/

of generating suitable trajectories while taking into consideration both the vehicular and environmental constraints. Given that the actual positions and relative pose of both DS and AUV, as well as the current disturbance, obstacle locations and no fly-zone boundaries are all affected by uncertainty, the proposed guidance system must have the ability to recompute trajectories very fast during the docking process.

To date, several attempts have been documented for the underwater homing and docking of AUVs [1–5]. A control law based on range-only measurements is developed in [6] for robust homing of the MARES AUV. This law results in asymptotical convergence of vehicle to a reference point and is validated theoretically using the Lyapunov theory. In [7], the Lyapunov stability theory is utilized for designing a guidance controller that generates the reference heading and crabbing angle to compensate for horizontal and vertical deviation during pool testing of docking operations. The experimental results showed that the system achieved 80% successful docking rate. Other classic guidance laws such as, pure pursuit guidance [8] and linear terminal guidance [9] are also considered for guiding an AUV towards a DS. Relying on a series of assumptions and simplifications, these laws try to minimize the drift and miss distance during the terminal phase. Even though these latter approaches are relatively simple to implement, they are limited in that they usually work in a controlled operating environment and operate based only on the geometric relationship and the AUV's kinematics. Neither of these approaches can provide a closed-form solution assuring a collision-free unsaturated-control motion. Satisfaction of the terminal conditions is at the mercy of the limitations of the components of the final speed and acceleration, or a fixed-time arrival. These approaches might be useful at the very last stages of docking operations when an AUV is within reach and aligned with the DS, however, arriving to this point should use a different approach (which might involve using other trajectory-shape-varying missile guidance algorithms). In [10], a fuzzy logic guidance system is employed for the docking purpose and shows a remarkable flexibility in terms of heading generation with respect to the relative range and bearing of the vehicle to the funnel-shaped DS considering the current disturbances.

It is noteworthy to mention that in all aforementioned approaches relatively little attention was paid to minimization of the energy expenditure or time consumed by a vehicle during docking operations. In this regard, a number of studies use the optimal control theory framework for an efficient AUV trajectory optimization aiming at minimizing some performance criteria such as time or energy. For instance, in [11] the NEROV vehicle's trajectory planning and collision avoidance scheme, that incorporate time and energy minimization, is developed using the optimal control theory framework; a numerical solution is then provided using the nonlinear programming package E04VCF. In [12], the optimal control theory framework is applied to find a time-optimal trajectory for a fully actuated AUV subject to actuation constraints. In [13], an efficient trajectory is generated based on developing a control strategy that minimizes the energy consumption along the desired path. The limitations of the thrusters are taken into account to produce an implementable trajectory for a real AUV. As a further example, [14] presents an energy minimization trajectory planning for a stable AUV using an analytically derived relationship between the energy consumption and number of thrusters.

Obtaining an analytical optimal solution for most aerospace and underwater guidance problems is typically very difficult if not impossible. Instead, an optimal control problem (OCP) is usually solved numerically. The Pontryagin's Maximum Principle serves as a foundation for indirect methods in which the optimal solution is synthetized by maximizing the Hamiltonian function. With the synthetized controls a two-point boundary-value-problem (TPBVP) is usually solved using one of the shooting methods. This approach faces several practical difficulties such as finding the initial guesses for co-states, non-converged solutions, and very-low computational efficiency particularly when problem dimensions are large. As a result, the indirect-method-based approach may not work in real-time applications where a host computing platform on a vehicle has a limited computation performance [15]. Direct methods, on the other hand, have attracted a great deal of attention due to their potency and properness for solving complex OCPs [15–18]. The essence of direct methods is to transform the original OCP into a finite nonlinear programming (NLP) problem by discretizing and parameterizing all the states and controls (or a sub-set of them). This results in near-optimal or quasi-optimal solutions that are suitable enough for the purpose of real-time trajectory generation when the overall system accuracy is overshadowed by uncertainties of operational environment, measurement errors, malfunctions of sensor suites, and possible limitations due to the vehicle's kinematics [18].

This paper formulates the TPBVP for the underwater docking problem and then employs the well-known direct method, inverse dynamics in the virtual domain (IDVD) method, to develop and test a real-time trajectory generator for realization on board of an AUV that is currently under the development at the Flinders University. This method has already been implemented on aerial [16,18–22], maritime [4] and space vehicles [23,24], and is now evaluated for the AUV dynamic model featuring seven states and three controls. The IDVD's effectiveness stems from several important properties

- parameterization of a candidate trajectory using a reduced number of states with unconditional satisfaction of the boundary conditions comprised of higher-order derivatives;
- utilization of the differential flatness property of the system's dynamics allowing the remaining states and controls to be expressed via aforementioned parameterizations rather than optimizing them, and decoupling of path optimization;
- optimization of the speed profile along the path.

The IDVD method is not subjected to the curse of dimensionality, does not need differentiability of the objective function and therefore any model and objective function can be utilized, and the generated controls are smooth and physically realizable. The IDVD method utilizes only a few variable parameters, features low sensitivity to variations of boundary conditions (BCs), and therefore guarantees a fast convergence rate of optimization routine. To achieve all these properties, the IDVD method gives up some optimality meaning that the value of the performance index is typically slightly higher than the one that could be achieved for the truly optimal solution satisfying the first-order optimality conditions. To benchmark the IDVD solution, the TPBV problem formulated in this paper is also solved offline, using another direct method, LGLPS method [25].

The paper is organized as follows. Section 2 introduces the Flinders' AUV and the software-in-the-loop (SITL) simulation environment developed to evaluate guidance, navigation and control algorithms prior to implementing them on board of a real vehicle. This section also introduces the AUV's seven-state model and formulates the TPBV problem for the underwater docking operations in a cluttered environment. Section 3 proceeds with using the Pontryagin's Maximum Principle to synthetize the optimal control structure. Analysis of this optimal control results in enhancing the original AUV model with three new states to allow satisfaction of the constraints imposed on the second-order derivatives of the original states for smooth departure and arrival of the AUV to the DS. This section also formally introduces the LGLPS method and discusses numerical solutions obtained using this method for a simple operational environment offline for both 7- and 10-state dynamics. Sections 4 and 5 are the core sections reintroducing the

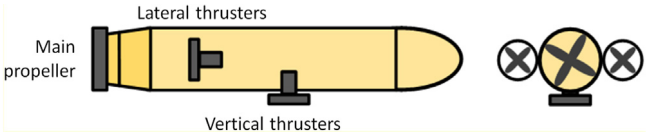

Fig. 1. Starboard view and rear view of Flinders AUV showing thruster locations.

IDVD method as applied to the AUV docking operations (Section 4) and thoroughly evaluating it for future application on a real AUV (Section 5). These evaluations include analysis of the IDVD solution by itself, comparison with the LGLPS solutions, and robustness tests involving SITL simulations, Monte Carlo trials, and realistic operational environment. The paper ends with the conclusion section.

2. Problem definition

This section describes the AUV that is currently under development at the Flinders University and the AUV Simulator already created to test guidance, navigation and control algorithms in the software-in-the-loop simulation environment. The section then goes on to present the formal mathematical formulation of the AUV docking problem.

2.1. Proposed control architecture and requirements

Fig. 1 depictures a controlled scheme of the AUV that is currently under development at Flinders University. This AUV features a typical torpedo-shape cylindrical body with ellipsoidal head and conical tail with about 120-cm total length and 20-cm diameter. The control scheme shown in Fig. 1 represents a portion of the overall control scheme envisioned to allow AUV maneuvering at low speeds to enforce precise operations. Three actuators enable independent motion in surge (bow/stern), heave (up/down) and yaw (side-to-side) directions. Specifically, a main propeller provides a forward motion, two lateral thrusters yield the yaw control and two vertical thrusters enable the depth control. The traditional submarine control surfaces, rudder and stern planes (not shown in Fig. 1), that serve as the primary controls at the high speeds, are still used in low-speed operations assuring roll and pitch stabilization. As a result, the yaw and heave motions for the configuration shown in Fig. 1 are decoupled from roll and pitch motion.

By design, the AUV control configuration shown in Fig. 1 assures fast and more accurate AUV repositioning and horizontal reorientation during slow-speed operations as required for a variety of missions including the docking operations. This particular research deals with docking with a stationary funnel-shaped DS requiring a unidirectional approach at a very low speed with a small possible variation in the heading angle (Fig. 2). The specific DS parameters used in this study were adopted from [26].

Typically, the docking operation includes two major phases. The first phase involves bringing the AUV close to the DS (several AUV lengths away from) and more or less aligned with the entrance cone centerline. The second phase comprises the straight-in approach with small corrections to keep proper alignment commences. The goal of this research effort is to develop an onboard trajectory generator capable of producing spatial trajectories combining both phases together and assuring a smooth arrival of AUV to the DS.

The overall control architecture for the docking operations is shown in Fig. 3 as realized within the SITL simulation environment that has been developed at the Flinders University to test and evaluate different guidance strategies for the Flinders AUV. This simulation environment incorporates realistically modeled

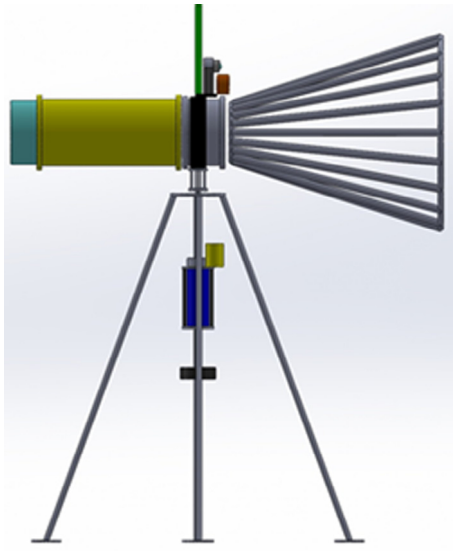

Fig. 2. Side view of stationary docking station.

components of the vehicle and emulates a real vehicle behavior while operating in a maritime environment [27–29]. As shown in Fig. 3, the SITL simulation environment includes three major components. First, it includes the high-fidelity AUV model adopted from [30] featuring fully-coupled six-degree-of-freedom dynamics of the vehicle and test-verified hydrodynamics coefficients including linear and nonlinear drag and lift, hydrodynamic added mass and inertia, and Coriolis force. It also includes the first-order-dynamics controller models and state estimation block. On top of that it allows emulating additional sensors providing situational awareness (SA) information. In the context of this research, SA information includes the updates about the relative position of DS and potential threats (mines, nets, kelp forest, sunken-ship debris) that need to be avoided on the way to the DS.

Second, the SITL simulation environment provides a testbed to test and verify different trajectory generation engines. This Trajectory generator module is required to be able to generate feasible and trackable reference trajectories, $\mathbf{x}^*(t)$, based on the mission goals, known or predicted operation environment. In the case of the mission goals change or change in perception of environmental elements and events with respect to time or space (triggered by the SA sensors or growing discrepancies between the current and expected AUV state) a new trajectory should be generated. Since a trajectory generator usually produces a trajectory that has a relatively small number of not-evenly-spaced (in time) nodes an Interpolator might be needed to adapt to the actual rate of a controller and produces an evenly-spaced set of M nodes, $\mathbf{x}(t_j)$, $j = 1, \ldots, M$.

The estimate of the current state at the time instant t_k, $\hat{\mathbf{x}}(t_k)$, and the interpolated reference trajectory are passed into the Controller module. This module mimics the controller that is intended to be used on board the Flinders AUV. This controller is based on the traveling waypoint (WP) guidance that computes a synthetic WP [31], which travels along the reference trajectory generated by the Trajectory generator module. In this specific implementation, the location of the traveling WP, $\mathbf{x}^{ref}(t_k)$, is set to be one meter ahead of $\hat{\mathbf{x}}(t_k)$ along $\mathbf{x}(t_j), j = 1, \ldots, M$. This is achieved by creating a 1 m radius sphere around the vehicle (corresponding to approximately 80% of the vehicle length) and then generating reference states based on the intersection of the sphere with the interpolated states. The difference $\delta\mathbf{x}(t_k) = \mathbf{x}^{ref}(t_k) - \hat{\mathbf{x}}(t_k)$ is fed into the sliding-mode controller (SMC). The SMC converts this difference into the

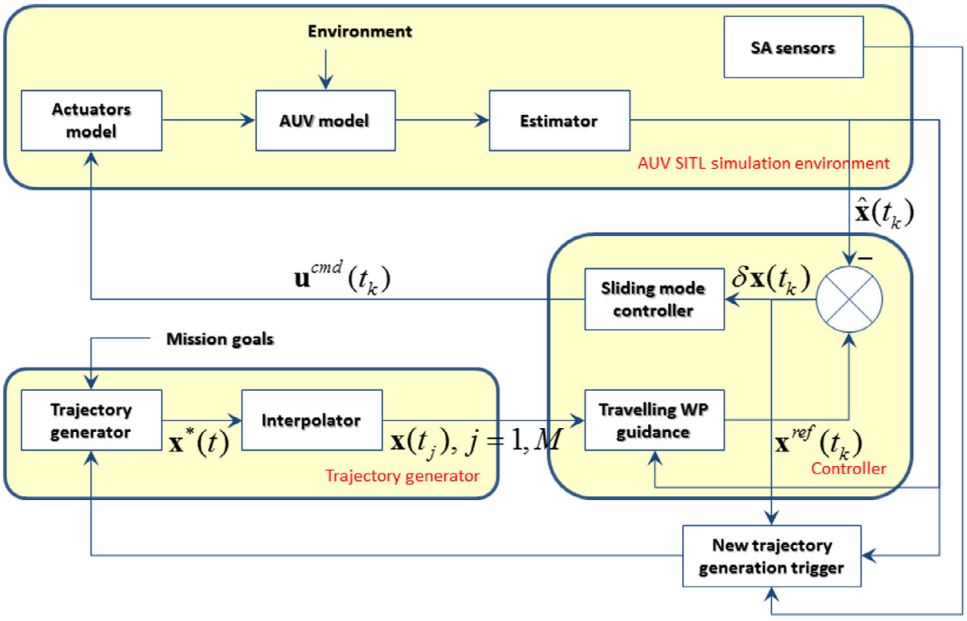

Fig. 3. The SITL simulation environment for the Flinders AUV.

commanded controls, $\mathbf{u}^{ref}(t_k)$, necessary to achieve $\mathbf{x}^{ref}(t_k)$. The details of SMC can be found in [28].

The proposed control architecture for the AUV docking operations establishes the following sets of requirements. The solution produced by the trajectory generator should

(1) Satisfy BCs on states and their derivatives
(2) Assure smooth departure from the current state and arrival into DS
(3) Obey state and control constraints
(4) Allow three-dimensional obstacle avoidance
(5) Account for AUV dynamics
(6) Feature smooth (differentiable) time histories of states and controls
(7) Take fraction of a second to compute on on-board computer.

On top of this, we would like the reference trajectory to optimize some performance index, such as time of maneuver, control actions expenditure, etc. In this particular application, we will require conservation of control actions while executing a fixed-time maneuver (meaning the docking time is negotiated upfront and fixed).

2.2. Mathematical formulation

Usually, the trajectory optimization problem deals with the simplest vehicle model possible. In our case, such a model would include translational and rotational kinematic equations

$$\begin{aligned} \dot{x} &= u\cos(\psi) + c_x \\ \dot{y} &= u\sin(\psi) + c_y \\ \dot{z} &= w \end{aligned} \tag{1}$$

$$\dot{\psi} = r \tag{2}$$

where x, y, and z are the coordinates of the AUV's center of gravity in the local tangential frame $\{n\}$; u and w are surge and heave velocity components relative to the water in the body frame $\{b\}$; ψ is the yaw angle; c_x and c_y are the x- and y-components of the current velocity vector ($|c_x| < u$, $|c_y| < u$). The origin of the North-East-Down frame $\{n\}$ is set at the surface, so that z represents the vertical distance from the surface, i.e. AUV depth.

However, pursuing a goal of producing a feasible solution (i.e. satisfying Requirement 5 formulated in Section 2.1) we should also include translational and rotational dynamics equations

$$\begin{aligned} m\dot{u} - (X_u + X_{u|u|}|u|)u &= T_u \\ m\dot{w} - (Z_w + Z_{w|w|}|w|)w &= T_w \end{aligned} \tag{3}$$

$$I_z\dot{r} - (N_r + N_{r|r|}|r|)r = T_r \tag{4}$$

where X_u, Z_w and N_r are the linear drag terms; $X_{u|u|}$, $Z_{w|w|}$, and $N_{r|r|}$ are the quadratic drag terms; m represents the mass of AUV and I_z is its inertia around the z axis (the values for these parameters were adopted from [32,33]). Finally, T_u, T_w, and T_r are the control inputs in surge, heave and yaw directions, respectively.

Hence, from the standpoint of trajectory optimization the state vector is represented by $\mathbf{x} = [x, y, z, \psi, u, w, r]^T$ and the control vector by $\mathbf{u} = [T_u, T_w, T_r]^T$. In this case, meeting Requirement 1 means satisfying the following BCs (at $t = 0$ and $t = t_f$):

$$\begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}_{t=0} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \psi_0 \end{bmatrix}, \quad \begin{bmatrix} u \\ w \\ r \end{bmatrix}_{t=0} = \begin{bmatrix} u_0 \\ w_0 \\ r_0 \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}_{t=t_f} = \begin{bmatrix} x_f \\ y_f \\ z_f \\ \psi_f \end{bmatrix}, \quad \begin{bmatrix} u \\ w \\ r \end{bmatrix}_{t=t_f} = \begin{bmatrix} u_f \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

The first order-derivatives of the states at the starting and terminal points then become

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix}_{t=0} = \begin{bmatrix} u_0\cos(\psi_0) + c_x \\ u_0\sin(\psi_0) + c_y \\ w_0 \\ r_0 \end{bmatrix}, \quad \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{r} \end{bmatrix}_{t=0} = \begin{bmatrix} m^{-1}(X_u + X_{u|u|}|u_0|)u_0 + m^{-1}T_{u;0} \\ m^{-1}(Z_w + Z_{w|w|}|w_0|)w_0 + m^{-1}T_{w;0} \\ I_z^{-1}(N_r + N_{r|r|}|r_0|)r_0 + I_z^{-1}T_{r;0} \end{bmatrix} \tag{7}$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\psi}\end{bmatrix}_{t=t_f}=\begin{bmatrix}u_f\cos(\psi_f)+c_x\\ u_f\sin(\psi_f)+c_y\\ 0\\ 0\end{bmatrix},\quad \begin{bmatrix}\dot{u}\\ \dot{w}\\ \dot{r}\end{bmatrix}_{t=t_f}=\begin{bmatrix}m^{-1}(X_u+X_{u|u|}|u_f|)u_f+m^{-1}T_{u;f}\\ m^{-1}T_{w;f}\\ I_z^{-1}T_{r;f}\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}. \tag{8}$$

Meeting Requirement 2 means ensuring

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\\ \ddot{\psi}\end{bmatrix}_{t=0}=\begin{bmatrix}\dot{u}_0\cos(\psi_0)-u_0r_0\sin(\psi_0)\\ \dot{u}_0\sin(\psi_0)+u_0r_0\cos(\psi_0)\\ \dot{w}_0\\ \dot{r}_0\end{bmatrix}$$
$$=\begin{bmatrix}(m^{-1}(X_u+X_{u|u|}|u_0|)u_0+m^{-1}T_{u;0})\cos(\psi_0)-u_0r_0\sin(\psi_0)\\ (m^{-1}(X_u+X_{u|u|}|u_0|)u_0+m^{-1}T_{u;0})\sin(\psi_0)+u_0r_0\cos(\psi_0)\\ m^{-1}(Z_w+Z_{w|w|}|w_0|)w_0+m^{-1}T_{w;0}\\ I_z^{-1}(N_r+N_{r|r|}|r_0|)r_0+I_z^{-1}T_{r;0}\end{bmatrix} \tag{9}$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\\ \ddot{\psi}\end{bmatrix}_{t=t_f}=\begin{bmatrix}(m^{-1}(X_u+X_{u|u|}|u_f|)u_f+m^{-1}T_{u;f})\cos(\psi_f)\\ (m^{-1}(X_u+X_{u|u|}|u_f|)u_f+m^{-1}T_{u;f})\sin(\psi_f)\\ m^{-1}T_{w;f}\\ I_z^{-1}T_{r;f}\end{bmatrix}$$
$$=\begin{bmatrix}0\\0\\0\\0.\end{bmatrix} \tag{10}$$

Resolving Eqs. (9) and (10) with respect to the controls yields

$$\begin{bmatrix}T_u\\ T_w\\ T_r\end{bmatrix}_{t=0}=\begin{bmatrix}m(\ddot{x}_0\cos(\psi_0)+\ddot{y}_0\sin(\psi_0))-(X_u+X_{u|u|}|u_0|)u_0\\ m\ddot{z}_0-(Z_w+Z_{w|w|}|w_0|)w_0\\ I_z\ddot{\psi}_0-(N_r+N_{r|r|}|r_0|)r_0\end{bmatrix} \tag{11}$$

$$\begin{bmatrix}T_u\\ T_w\\ T_r\end{bmatrix}_{t=t_f}=\begin{bmatrix}-(X_u+X_{u|u|}|u_f|)u_f\\ 0\\ 0\end{bmatrix}. \tag{12}$$

Eq. (12) defines the control values at $t = t_f$ required for the smooth arrival (enforcing the last equations in (8) and (10)) and Eq. (11) defines the control values at $t = 0$ required for the smooth departure from the initial conditions (which is especially important when the reference trajectory needs to be recomputed while executing the previously generated solution).

Obeying state and control constraints (Requirement 3) implies

$$|r|\le r^{\max},\quad 0<z\le z^{\max} \tag{13}$$

$$|T_u|\le T_u^{\max},\quad |T_w|\le T_w^{\max},\quad |T_r|\le T_r^{\max} \tag{14}$$

and avoiding the three-dimensional no-fly areas (obstacles) (Requirement 4), in the case when they can be represented as spheres with radii $R_l\,(l=1,\ldots,L)$ located at $[x_l,y_l,z_l]^T$, means ensuring

$$(x-x_l)^2+(y-y_l)^2+(z-z_l)^2\ge R_l^2. \tag{15}$$

From the optimal control point of view, the trajectory optimization problem for driving an AUV into a DS can be formulated in the form of TPBVP as follows. Starting from $\mathbf{x}_0=[x_0,y_0,z_0,\psi_0,u_0,w_0,r_0]^T$ with $\mathbf{u}_0=\left[T_{u;0},T_{w;0},T_{r;0}\right]^T$ at $t=0$ we need to bring the AUV to the DS with $\mathbf{x}_f=\left[x_f,y_f,z_f,\psi_f,u_f,w_f,r_f\right]^T$ and $\mathbf{u}_f=\left[T_{u;f},T_{w;f},T_{r;f}\right]^T$ at $t=t_f$ while obeying constraints (1)–(4) and (11)–(12) and minimizing the sum of the squared control inputs over the fixed time interval $T=t_f$, so that the normalized performance index becomes

$$J=\frac{1}{T(T_u^{\max})^2}\int_0^T(T_u^2+T_w^2+T_r^2)dt. \tag{16}$$

With this performance index the quantity $100(1-\sqrt{J})\%$ provides an estimate of the saved control input expenditure compared to the case when the maneuver is performed at control bounds ($T_u(t)=T_u^{\max}$).

3. Obtaining benchmark optimal solution

This section uses the Pontryagin's Maximum Principle to synthetize the optimal controls. The LGLPS method is then used to obtain a benchmark numerical solution. Since this solution happens to be not feasible, the original 7-state system (1)–(4) is then expanded to the 10-state system. This allows enforcing boundary constraints on the second-order derivatives of AUV coordinates. Even though the LGLPS-method-based solutions are obtained offline (the LGLPS method requires a lot of computational resources and does not necessarily guarantee convergence), it can still serve as a benchmark solution to assess the optimality of other solutions.

3.1. Synthesis of the optimal control

Following the optimal control theory the Hamiltonian for the system (1)–(4) with the performance index (16) can be written as [34]

$$\begin{aligned}H=&-T^{-1}(T_u^{\max})^{-2}(T_u^2+T_w^2+T_r^2)+\lambda_x(u\cos(\psi)+c_x)\\&+\lambda_y(u\sin(\psi)+c_y)+\lambda_z w+\lambda_\psi r\\&+\lambda_u(X_u u+X_{u|u|}\,|u|\,u+T_u)m^{-1}\\&+\lambda_w(Z_w w+Z_{w|w|}\,|w|\,w+T_w)m^{-1}\\&+\lambda_r(N_r r+N_{r|r|}\,|r|\,r+T_r)I_z^{-1}=const\end{aligned} \tag{17}$$

with λ_x, λ_y, λ_z, λ_u, λ_w, λ_ψ, and λ_r being co-state variables.

The differential equations for co-state variables can be written as follows

$$\begin{aligned}\dot{\lambda}_x&=0\\ \dot{\lambda}_y&=0\\ \dot{\lambda}_z&=0\end{aligned} \tag{18}$$

$$\dot{\lambda}_\psi=\lambda_x u\sin(\psi)-\lambda_y u\cos(\psi) \tag{19}$$

$$\begin{aligned}\dot{\lambda}_u&=-\lambda_x\cos(\psi)-\lambda_y\sin(\psi)-\lambda_u(X_u+X_{u|u|}\,|u|)m^{-1}\\ \dot{\lambda}_w&=-\lambda_z-\lambda_w(Z_w+Z_{w|w|}\,|w|)m^{-1}\\ \dot{\lambda}_r&=-\lambda_\psi-\lambda_r(N_r+2N_{r|r|}\,|r|)I_z^{-1}.\end{aligned} \tag{20}$$

According to the Pontryagin's Maximum Principle the optimal control structure is obtained by differentiating (17) with respect to the control inputs and equating the derivatives to zero, which results in

$$\begin{bmatrix}T_u\\ T_w\\ T_r\end{bmatrix}_{opt}=\frac{T(T_u^{\max})^2}{2}\begin{bmatrix}\lambda_u m^{-1}\\ \lambda_w m^{-1}\\ \lambda_r I_z^{-1}\end{bmatrix}. \tag{21}$$

These are the optimal control values as long as they fall within the limits (14).

Unfortunately, the augmented system (1)–(4), (18)–(20) has no analytical solution and trying to solve it numerically by varying the initial values of co-states is very problematic as well, not to mention that accounting for constraints (13) and (15) dictates replacing Eq. (17) with the augmented Hamiltonian

$$\overline{H}=H+\boldsymbol{\mu}^T\mathbf{g}(\mathbf{x},\mathbf{u})=const \tag{22}$$

which makes a problem even more difficult to solve (in Eq. (22) $\mathbf{g}(\mathbf{x},\mathbf{u})$ represents the vector of constrains (13) and (15) with μ being the vector of Lagrangian multipliers). On top of the control

structure equation (21) allows a singular control arc when $\lambda_r \equiv 0$ (singular control arc complicates a numerical solution even further). This implies $\dot{\lambda}_r \equiv 0$ and from the last equation in (20) $\lambda_\psi = 0$. The latter leads to $\dot{\lambda}_\psi \equiv 0$ in Eq. (19) and therefore

$$\lambda_x \sin(\psi) \equiv \lambda_y \cos(\psi). \tag{23}$$

Knowing that λ_x and λ_y are constants (see Eq. (18)), equality (23) results in $\psi \equiv const$ and therefore $\dot{\psi} \equiv 0$, $r \equiv 0$ (from Eq. (2)), $\dot{r} \equiv 0$, and $T_{r;opt} \equiv 0$ (from Eq. (4)). Hence, a singular arc represents a straight motion.

Now, even if the augmented system (1)–(4), (18)–(20) could somehow be solved it would still not meet Requirement 2 expressed mathematically in Eqs. (11) and (12). In order to satisfy BCs on controls, we need to convert them into the new states and use the new controls. As a result, the original system (1)–(4) should be augmented with

$$\begin{aligned} \dot{T}_u &= \upsilon_u \\ \dot{T}_w &= \upsilon_w \\ \dot{T}_r &= \upsilon_r. \end{aligned} \tag{24}$$

That also implies concluding constraints (14) into the augmented Hamiltonian (22). The bounds on the new controls would be established as

$$|\upsilon_u| \le \upsilon_u^{\max}, \quad |\upsilon_w| \le \upsilon_w^{\max}, \quad |\upsilon_r| \le \upsilon_r^{\max}. \tag{25}$$

The Hamilton (17) will now have three additional terms

$$H^* = H + \lambda_{Tu}\upsilon_u + \lambda_{Tw}\upsilon_w + \lambda_{Tr}\upsilon_r = const \tag{26}$$

with three more differential equations for the new co-states

$$\begin{aligned} \dot{\lambda}_{Tu} &= 2T^{-1}(T_u^{\max})^{-2}T_u - \lambda_u m^{-1} \\ \dot{\lambda}_{Tw} &= 2T^{-1}(T_u^{\max})^{-2}T_w - \lambda_w m^{-1} \\ \dot{\lambda}_{Tr} &= 2T^{-1}(T_u^{\max})^{-2}T_r - \lambda_r I_z^{-1}. \end{aligned} \tag{27}$$

Differentiating Eq. (26) with respect to the three new controls yields the bang–bang optimal control structure

$$\begin{bmatrix} T_u \\ T_w \\ T_r \end{bmatrix}_{opt} = \begin{bmatrix} \upsilon_u^{\max} sign(\lambda_{Tu}) \\ \upsilon_w^{\max} sign(\lambda_{Tw}) \\ \upsilon_r^{\max} sign(\lambda_{Tr}) \end{bmatrix} \tag{28}$$

with a possibility of a singular control arc when $\lambda_{Tu} \equiv 0$, $\lambda_{Tw} \equiv 0$, and $\lambda_{Tr} \equiv 0$.

3.2. Basics of the LGLPS method

It is well known that obtaining numerical solutions for the TPBVP like the one presented in the previous section is quite challenging. In the past decade, however, several pseudo-spectral (PS) methods were successfully deployed to solve complex OCPs offline [35–37]. The general idea behind PS methods, being the direct method, is to transcribe an OCP into a NLP relying on a certain non-uniform computational node distribution to compute a quadrature defined by a performance index. Specifically, the LGLPS method transcribes the TPBVP of Section 2.2 into the NLP problem by parameterizing state and control time histories, $\mathbf{x}(\bar{t})$ and $\mathbf{u}(\bar{t})$, over the N Gauss–Lobatto collocation nodes using the orthogonal Lagrange interpolating polynomials $\varphi_i(\bar{t})$ [37,38]

$$\mathbf{x}(\bar{t}) = \sum_{i=0}^{N} \mathbf{x}(\bar{t}_i)\,\varphi_i(\bar{t}), \quad \mathbf{u}(\bar{t}) = \sum_{i=0}^{N} \mathbf{u}(\bar{t}_i)\,\varphi_i(\bar{t}) \tag{29}$$

where $\bar{t} = (2t - t_f - t_0)(t_f - t_0)^{-1} \in [-1; 1]$ is the scaled time and

$$\varphi_i(\bar{t}) = \frac{1}{N(N+1)L_N(\bar{t})} \frac{(\bar{t}^2 - 1)\dot{L}_N(\bar{t})}{\bar{t} - \bar{t}_i}. \tag{30}$$

In Eq. (30) $\dot{L}_N(\bar{t})$ represents differentiation of $L_N(\bar{t})$ with respect to time, where $L_N(\bar{t})$ denotes the Legendre polynomial of order N

$$L_N(\bar{t}) = \frac{1}{2^N N!} \frac{d^N}{d\bar{t}} (\bar{t}^2 - 1)^N. \tag{31}$$

The LGLPS method utilizes the same LGL nodes for discretization and collocation computed as

$$\bar{t}_i = \begin{cases} -1 & \text{for } i = 0 \\ i_{th} \text{ root of } \dot{L}_N(\bar{t}) & \text{for } i = 1, 2, \ldots, N-1 \\ 1 & \text{for } i = N. \end{cases} \tag{32}$$

System dynamics (1)–(4) are enforced by imposing the corresponding constraints at the LGL nodes. Specifically, differential equations describing system's behavior are transcribed into the algebraic equations using collocation procedure

$$\begin{aligned} &\dot{\mathbf{x}}(\bar{t}_k) = \sum_{i=0}^{N} \dot{L}_i(\bar{t}_i)\mathbf{x}(\bar{t}_i) = \sum_{i=0}^{N} D_{ki}\mathbf{x}(\bar{t}_i), \\ &\sum_{i=0}^{N} D_{ki}\mathbf{x}(\bar{t}_i) - \frac{t_f - t_0}{2}\,\mathbf{f}(\mathbf{x}(\bar{t}_i),\ \mathbf{u}(\bar{t}_i)) = \mathbf{0}, \quad k = 0, \ldots, N \end{aligned} \tag{33}$$

where $\mathbf{f}(\mathbf{x}(\bar{t}_i),\ \mathbf{u}(\bar{t}_i))$ represents the right-hand-side of the corresponding state equation, and D_{ki} is the $(N+1)$-by-$(N+1)$ differentiation matrix

$$D_{ki} = \begin{cases} \dfrac{L_N(\bar{t}_k)}{L_N(\bar{t}_i)} \dfrac{1}{(\bar{t}_k - \bar{t}_i)} & \text{if } k \ne i \\ -\dfrac{N(N+1)}{4} & \text{if } k = i = 0 \\ \dfrac{N(N+1)}{4} & \text{if } k = i = N \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

The BCs are imposed as equality constraints on the first and last collocation nodes, $\bar{t}_0 = -1$ and $\bar{t}_N = 1$, respectively. The performance index (16) is estimated using the Gaussian quadrature rule

$$\int_{-1}^{1} P(\bar{t})\,d\bar{t} = \sum_{k=0}^{N} P(\bar{t}_k) w_k, \quad k = 0, \ldots, N \tag{35}$$

where the quadrature weights w_k are given by

$$w_k = \frac{2}{N(N+1)} \frac{1}{(L_N(\bar{t}_k))^2}, \quad k = 0, \ldots, N. \tag{36}$$

The details on how the LGLPS method handles the first-order (Karush–Kuhn–Tucker) necessary conditions for a solution in nonlinear programming to be optimal can be found in [39].

The LGLPS method solution convergence depends on the initial guess, number of collocation nodes, sparsity pattern, optimization solver, and numerical precision of approximated solution.

3.3. Comparison of the 7- and 10-state solutions

Figs. 4–10 show an example of applying the LGLPS method to solve TPBVP (1)–(6), (13)–(14), (16) without and with satisfying Eqs. (11)–(12) or in other words using 7 states as in Eqs. (1)–(4) or 10 states as in Eqs. (11)–(12) plus Eq. (24). All computer simulations discussed in this paper were performed on a desktop computer with an Intel i7 3.40 GHz quad-core processor using MATLAB® R2015a development environment. The MATLAB *fmincon* function was used throughout this research as a solver (this function realizes a gradient-based method that is designed to work on problems where the objective and constraint functions are both continuous and have continuous first derivatives; other solvers that are usually used with PS methods are the sparse nonlinear optimizer (SNOPT), and interior point optimizer (IPOPT).

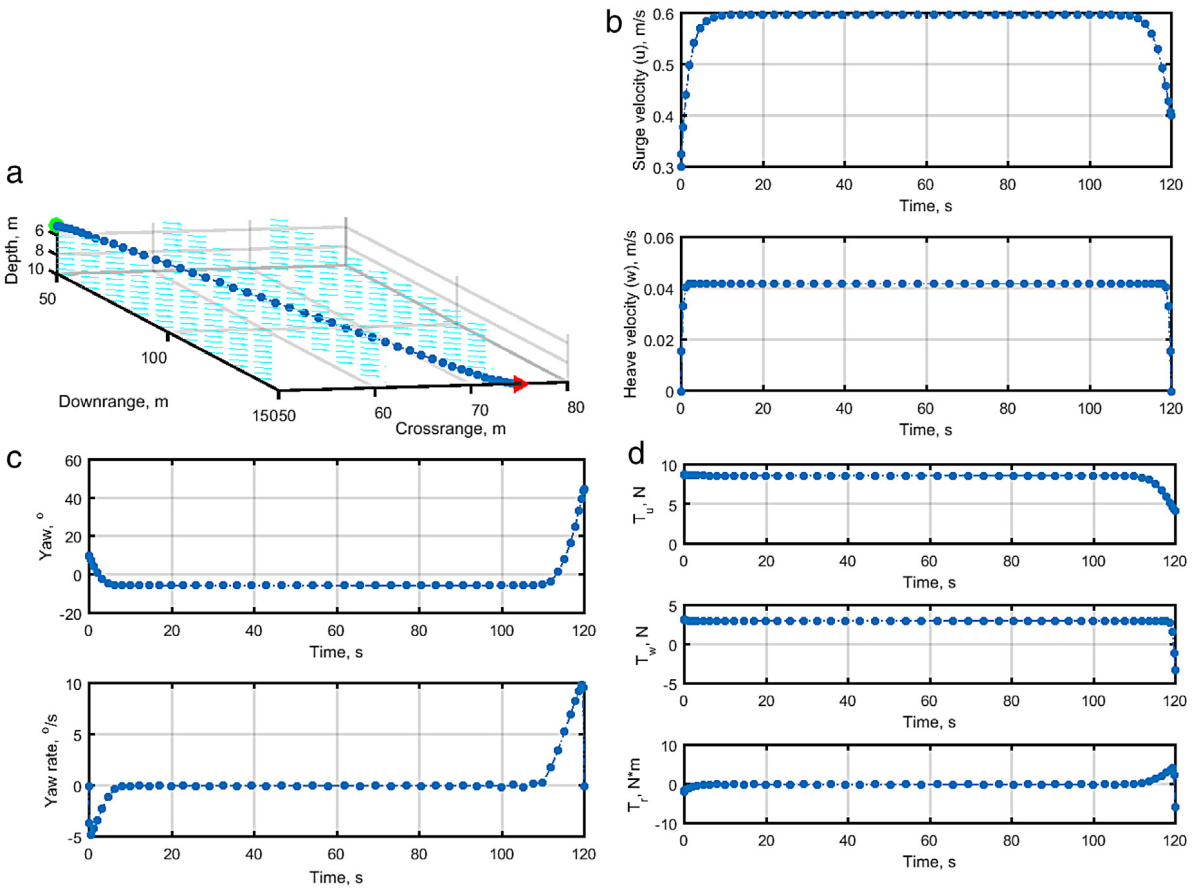

Fig. 4. Example of the 7-state solution.

The specific scenario was set as follows. The AUV is required to perform docking with a stationary DS, the pose of which is known a priori. There are no no-fly zones or obstacles (15). The operational environment features a known two-dimensional (2D) current with components $c_x = 0.25$ m/s and $c_y = 0.25$ m/s (Eq. (1)). The BCs are defined as

$$\mathbf{x}_0 = [50\text{ m}, 50\text{ m}, 5\text{ m}, 10^\circ, 0.3\text{ m/s}, 0\text{ m/s}, 0^\circ/\text{s}]^T \text{ and}$$
$$\mathbf{x}_f = [150\text{ m}, 75\text{ m}, 10\text{ m}, 45^\circ, 0.4\text{ m/s}, 0\text{ m/s}, 0^\circ/\text{s}]^T. \quad (37)$$

The AUV is characterized by $m = 30.5$ kg, $I_z = 3.45$ kg $\cdot$ m^2, $X_u = -13.5$ kg/s, $X_{u|u|} = -1.62$ kg/m, $Z_w = -66.6$ kg/s, $Z_{w|w|} = -131$ kg/m, $N_r = -6.87$ kg $\cdot$ m^2/s, $N_{r|r|} = -94$ kg $\cdot$ m^2 (3)–(4). The constraints defined by the physical limitations of AUV's actuators (14) and its yaw rate (13) are $T_u^{\max} = T_w^{\max} = 20$ N, $T_r^{\max} = 20$ N$\cdot$m, and $r^{\max} = 15^\circ$/s.

Fig. 4 demonstrates the numerical solution that was achieved using the LGLPS method based on the 7-state formulation and 50-node discretization. Specifically, Fig. 4(a) shows the three-dimensional (3D) view of the obtained solution with multiple arrows denoting the constant (and known) current, while Figs. 4(b)–(d) present time histories of velocity vector components (Fig. 4(b)), yaw angle and yaw rate (Fig. 4(c)) and three controls (Fig. 4(d)). As clearly seen from Fig. 4, the 7-state formulation does not allow accounting for comptroller dynamics (11)–(12) resulting in the instantaneous jumps in the state magnitudes and consequently –non-realizable control actions (Fig. 4(d)).

The optimality of the obtained solution can be assessed by observing time-histories of co-states (Fig. 5) and the Hamiltonian (Fig. 6). Fig. 5(a) indicates that differential equations (18) hold (in the numerical sense), and Fig. 5(c) indicates that most of the time the solution features a singular arc control (23). Oscillations (instability) in λ_x, λ_y, and λ_z, especially towards the ends of the trajectory, are due to the fact that the differential and integral forms of the LGLPS method are not equivalent resulting in the LGLPS discrete co-state matrix being rank-deficient; this causes co-states oscillation about the true solution and leading to potential non-converged co-states [40,41].

The Hamiltonian (17) is explicitly independent of time and therefore should be constant. Fig. 6 shows that it is the case along the singular arc. However, the optimization routine has difficulty in numerically solving the problem beyond that.

It should be noted that obtaining numerical solutions like the one shown in Figs. 4–6 is not an easy task. As mentioned in Section 3.2 the LGPLS approach requires a reasonable initial guess on time histories for all states and controls, and generally speaking starts producing a feasible solution for discretization involving more than 50 nodes [40]. With these 50 nodes for each of the seven co-states and three controls, the number of varied parameters is 500 and therefore solving the problem numerically requires substantial computational resources. Obviously, the LGPLS-based approach cannot be realized on board of the AUV for real-time operations, but can be used (when appropriate) to provide a point of reference against which other approaches may be assessed in terms of closeness to the "optimal" solution [40].

As mentioned in Section 3.2, the infeasibility of the optimal control solution for system (1)–(4) can potentially be fixed by expanding the original system to include three more equations converting the original controls to the three new states (24). The results of numerical solution for the same TPBVP with the 10-state problem formulation are shown in Figs. 7–10. In this realization, the bound on the new controls (25) were established as $\upsilon_u^{\max} = \upsilon_w^{\max} = 0.5$ N/s and $\upsilon_r^{\max} = 0.5$ N $\cdot$ m/s.

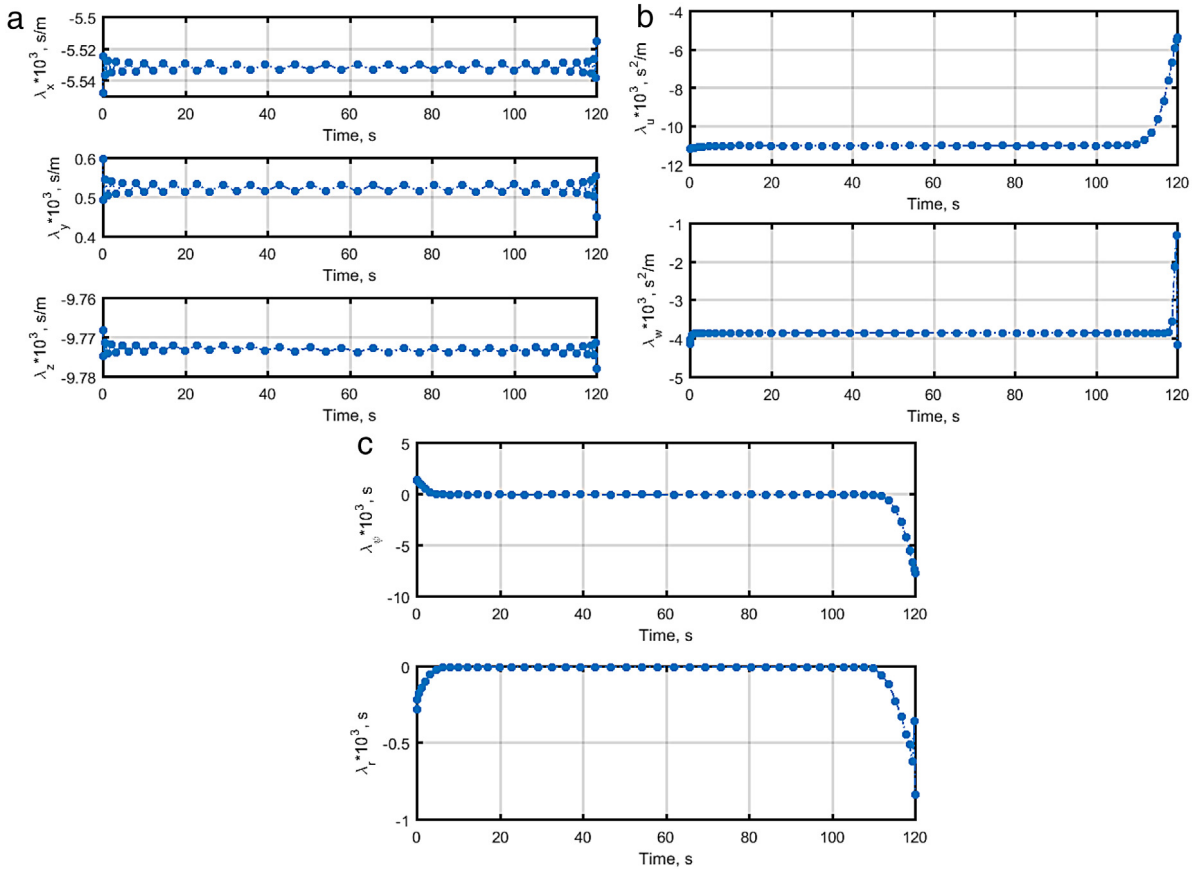

Fig. 5. Co-state time histories for the 7-state solution of Fig. 4.

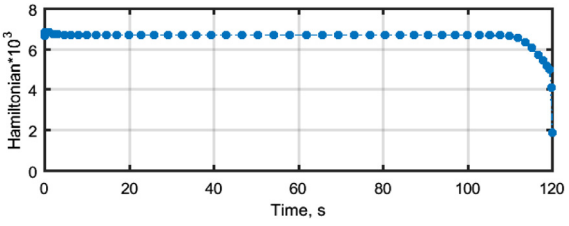

Fig. 6. The Hamiltonian for the 7-state solution of Figs. 4 and 5.

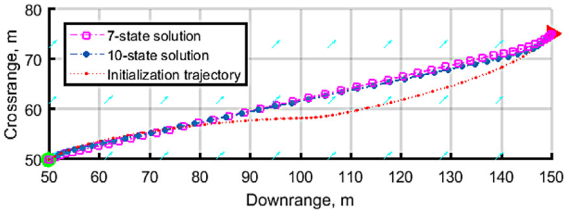

Fig. 7. Comparison of the 10-state and 7-state solutions.

Specifically, Fig. 7 shows a bird's view of the 10-state trajectory as compared to the 7-state solution. The major difference is in less-control-demanding departure from the initial point and arrival to the final point. Fig. 7 also shows the initial guess that was used to assure and speed up the convergence of both 7- and 10-state solutions (this solution was obtained using the real-time feasible trajectory generation method that is promulgated in this paper and discussed in the next section).

Fig. 8(a), (b) demonstrate a big difference in the beginning and at the end of the trajectory as compared to Fig. 4(b), (c). The BCs (11) and (12) can now be satisfied (Fig. 8(c)) assuring a smooth departure from the initial point and smooth arrival to DS.

The time histories of the new controls, υ_u, υ_w, and υ_r, are shown in Fig. 8(d). Indeed, according to Eqs. (27)–(28) they feature a bang-singular-bang optimal control.

Time histories for the co-states and Hamiltonian are shown in Figs. 9, 10. As was the case in the 7-state formulation (Fig. 6), the Hamiltonian for the 10-state solution (Fig. 10) is not constant, meaning that the optimality conditions in the approximate numerical solution are not necessarily satisfied.

4. Real-time trajectory optimization procedure

As shown in the previous section, it takes many efforts to try to solve the TPBVP, formulated in Section 2.2, using the LGLPS method. Strictly speaking, the solutions are not necessarily truly optimal and directly utilizable by relatively low power on-board computers. This section, introduces another approach, which uses the IDVD method to transcribe the TPBVP into a low-dimensional NLP problem thus sacrificing some optimality in order to achieve feasible trajectories in real time.

4.1. Concept of IDVD method

The first major principle of the IDVD method [16] is to use the differential flatness of system dynamics, to significantly reduce the dimension of the optimization problem and thus enable fast prototyping of feasible trajectories [18]. This is achieved by expressing all states and controls as the functions of a so-called output vector.

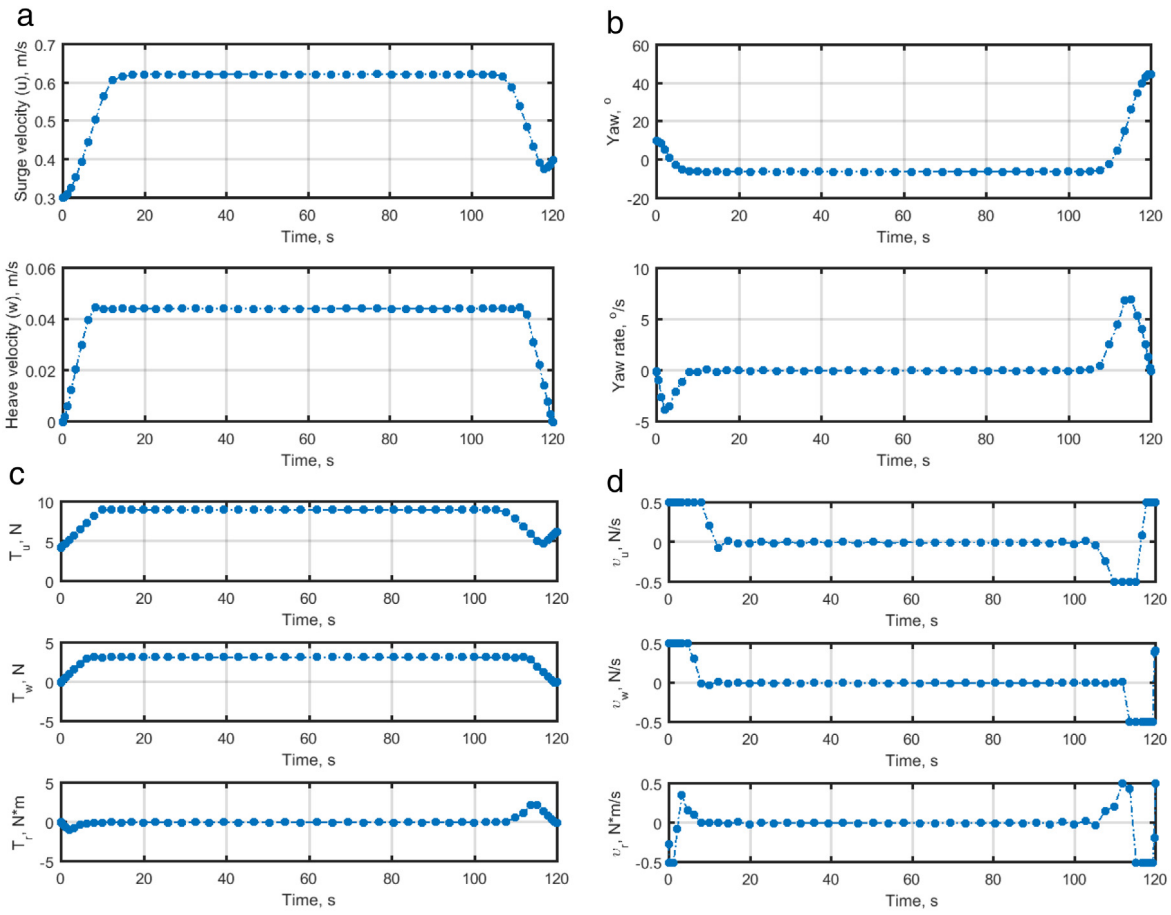

Fig. 8. State (a–c) and control (d) time histories of the 10-state solution.

For the specific system (1)–(4) we can invert Eq. (1) to get

$$\begin{aligned} u &= ((\dot{x}-c_x)^2+(\dot{y}-c_y)^2)^{0.5} \\ \psi &= \tan^{-1}\left(\frac{\dot{y}-c_y}{\dot{x}-c_x}\right) \\ w &= \dot{z}. \end{aligned} \tag{38}$$

Then, inverting Eq. (2) yields

$$r = \dot{\psi} \tag{39}$$

finally, Eqs. (3) and (4) can be rewritten as

$$\begin{aligned} T_u &= m\dot{u} - (X_u + X_{u|u|}\,|u|)u \\ T_w &= m\dot{w} - (Z_w + Z_{w|w|}\,|w|)w \\ T_r &= I_z\dot{r} - (N_r + N_{r|r|}\,|r|)r. \end{aligned} \tag{40}$$

Hence, if we define the output vector as $\mathbf{y} = [x, y, z]$, the original 7-state vector $\mathbf{x}$ can be expressed as

$$\mathbf{x} = \mathbf{f}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}) \tag{41}$$

while the control vector $\mathbf{u}$ becomes

$$\mathbf{u} = \mathbf{h}(\dot{\mathbf{y}}, \ddot{\mathbf{y}}). \tag{42}$$

As a result, for the specific formulation (1)–(4) we can reduce the required number of parameterizations from ten (seven states and three controls as was in the case of LGLPS method) to just three (cf. to Eq. (29)).

The second key principle of the IDVD method is to additionally decouple path and speed profile optimization, i.e. "break" kinematic equations (38). This is done by conducting all computations in the virtual domain characterized by the virtual arc τ rather than by time t. The mapping between the physical domain and the virtual domain of argument is performed using the so-called speed factor $\lambda(\tau)$

$$\lambda(\tau) = \frac{d\tau}{dt}. \tag{43}$$

Differentiation of any time-variant parameter ξ with respect to time can be expressed as

$$\dot{\xi} = \lambda\xi', \quad \ddot{\xi} = \lambda(\lambda'_\tau\xi'_\tau + \lambda\xi''_{\tau\tau}), \; etc. \tag{44}$$

(subscript τ means differentiation with respect to the virtual arc). When needed, relations (44) can be inverted to

$$\xi'_\tau = \lambda^{-1}\dot{\xi}, \quad \xi''_{\tau\tau} = \lambda^{-2}\ddot{\xi} - (\lambda'_\tau\lambda^{-1}\xi'). \tag{45}$$

Using Eq. (44), Eq. (38) becomes

$$\begin{aligned} u &= ((\lambda x'_\tau - c_x)^2 + (\lambda y'_\tau - c_y)^2)^{0.5} \\ \psi &= \tan^{-1}\left(\frac{\lambda x'_\tau - c_x}{\lambda y'_\tau - c_y}\right) \\ w &= \lambda z'_\tau \end{aligned} \tag{46}$$

and Eqs. (39), (40) result in

$$r = \lambda\psi'_\tau \tag{47}$$

$$\begin{aligned} T_u &= m\lambda u'_\tau - (X_u + X_{u|u|}\,|u|)u \\ T_w &= m\lambda w'_\tau - (Z_w + Z_{w|w|}\,|w|)w \\ T_r &= I_z\lambda r'_\tau - (N_r + N_{r|r|}\,|r|)r. \end{aligned} \tag{48}$$

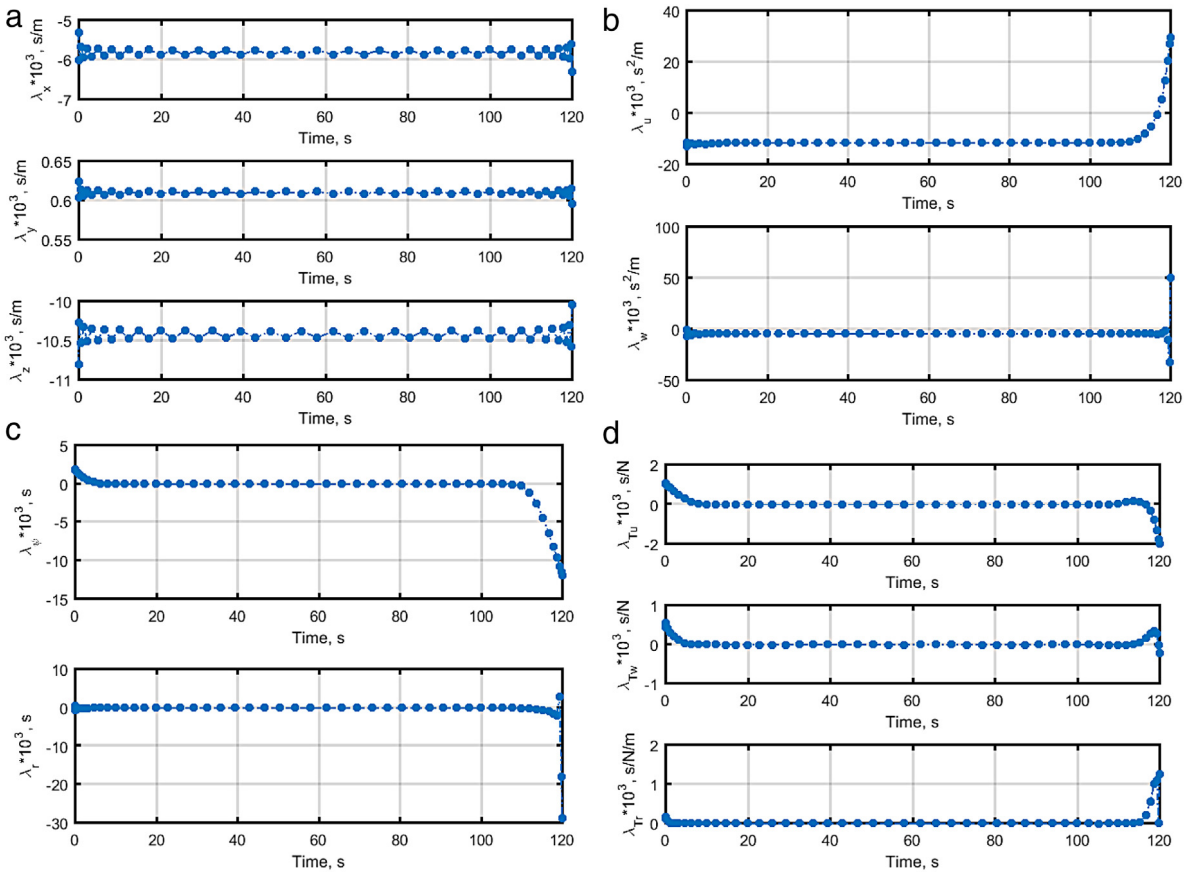

Fig. 9. Co-state time histories for the 10-state solution of Fig. 8.

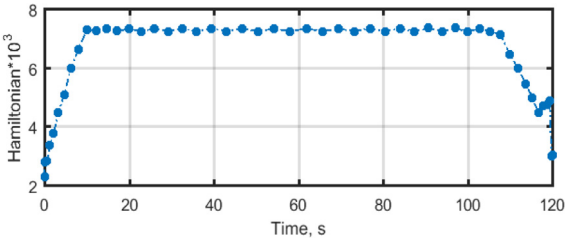

Fig. 10. The Hamiltonian for the 10-state solution of Figs. 8, 9.

Introducing $\lambda(\tau)$ allows moving along the path defined by the output vector **y** with any speed profile, hence enabling more optimization flexibility and controllability.

The third and final principle is to use a high-order parameterization for the components of the output vector and define all parameterization parameters by satisfying the BCs. A good analogue would be defining the coefficients of the third-order polynomial by using two control points (Bézier curves) rather than varying the coefficients of approximation themselves. This feature allows satisfying BCs (5)–(10) by default. More details on this subject are provided in the next section.

4.2. Parameterization of the output vector

In general, parameterization of the output vector can be based on a set of any reference (basis) functions of some abstract scaled argument $\bar{\tau} = \tau/\tau_f \in [0; 1]$ including any combination of monomial or trigonometric terms [16], as long as it can represent the expected general shape. On practice, even a 3rd-order-polynomial (spline) for each element of the output vector would suffice. However, with the 3rd-order polynomial, represented by four varied coefficients, we would be able to satisfy only four BCs (up to the first order derivative at the initial and terminal points). In general, the minimum order of parameterization (the number of terms in the case of using monomials) is determined via the number of BCs that should be satisfied [16]

$$n_p = d_0 + d_f + 1 \tag{49}$$

where d and d_f are the highest order of time derivatives at the initial and terminal points. In our case, we need to satisfy up to the second-order derivative (9)–(10), hence, parameterization should include at least two more terms. Instead of using the 5th order polynomial, we could also add two trigonometric terms, so that they become zeros at the boundaries of interval $\bar{\tau} \in [0; 1]$. This is a convenient way to vary the curvature of the reference trajectory at the initial and terminal points without causing oscillatory behavior as we might have with the higher order polynomials.

Hence, to address the TPBVP the three spatial coordinates x, y, z constituting the output vector may be simply parameterized as follows:

$$\begin{aligned} x(\bar{\tau}) &= P_x(\bar{\tau}) \\ &= a_{0x} + a_{1x}\bar{\tau} + a_{2x}\bar{\tau}^2 + a_{3x}\bar{\tau}^3 + b_{1x}\sin(\pi\bar{\tau}) + b_{2x}\sin(2\pi\bar{\tau}) \\ y(\bar{\tau}) &= P_y(\bar{\tau}) \\ &= a_{0y} + a_{1y}\bar{\tau} + a_{2y}\bar{\tau}^2 + a_{3y}\bar{\tau}^3 + b_{1y}\sin(\pi\bar{\tau}) + b_{2y}\sin(2\pi\bar{\tau}) \\ z(\bar{\tau}) &= P_z(\bar{\tau}) \\ &= a_{0z} + a_{1z}\bar{\tau} + a_{2z}\bar{\tau}^2 + a_{3z}\bar{\tau}^3 + b_{1z}\sin(\pi\bar{\tau}) + b_{2z}\sin(2\pi\bar{\tau}) \end{aligned} \tag{50}$$

where the coefficients $a_{i\eta}$ and $b_{i\eta}$ ($\eta = \{x, y, z\}$) in these parameterizations can be defined by BCs (5)–(10).

With the order of parameterization being exactly n_p as defined in Eq. (49), the only varied parameter in Eq. (50) is the virtual arc length τ_f. In order to have more flexibility to vary the shape of the candidate trajectory, more terms should be added. Again, we could add more trigonometric terms or more monomials. For example, we can add two more monomial terms

$$\begin{aligned}
x(\bar{\tau}) &= P_x(\bar{\tau}) = a_{0x} + a_{1x}\bar{\tau} + a_{2x}\bar{\tau}^2 + a_{3x}\bar{\tau}^3 + a_{4x}\bar{\tau}^4 \\
&\quad + a_{5x}\bar{\tau}^5 + b_{1x}\sin(\pi\bar{\tau}) + b_{2x}\sin(2\pi\bar{\tau}) \\
y(\bar{\tau}) &= P_y(\bar{\tau}) = a_{0y} + a_{1y}\bar{\tau} + a_{2y}\bar{\tau}^2 + a_{3y}\bar{\tau}^3 + a_{4y}\bar{\tau}^4 \\
&\quad + a_{5y}\bar{\tau}^5 + b_{1y}\sin(\pi\bar{\tau}) + b_{2y}\sin(2\pi\bar{\tau}) \\
z(\bar{\tau}) &= P_z(\bar{\tau}) = a_{0z} + a_{1z}\bar{\tau} + a_{2z}\bar{\tau}^2 + a_{3z}\bar{\tau}^3 + a_{4z}\bar{\tau}^4 \\
&\quad + a_{5z}\bar{\tau}^5 + b_{1z}\sin(\pi\bar{\tau}) + b_{2z}\sin(2\pi\bar{\tau}).
\end{aligned} \tag{51}$$

Differentiating Eq. (51) three times with respect to the $\bar{\tau}$ yields

$$\begin{aligned}
\tau_f\eta'_\tau &= a_{1\eta} + 2a_{2\eta}\bar{\tau} + 3a_{3\eta}\bar{\tau}^2 + 4a_{4\eta}\bar{\tau}^3 + 5a_{5\eta}\bar{\tau}^4 \\
&\quad + \pi b_{1\eta}\cos(\pi\bar{\tau}) + 2\pi b_{2\eta}\cos(2\pi\bar{\tau}) \qquad (52)\\
\tau_f^2\eta''_\tau &= 2a_{2\eta} + 6a_{3\eta}\bar{\tau} + 12a_{4\eta}\bar{\tau}^2 + 20a_{5\eta}\bar{\tau}^3 - \pi^2 b_{1\eta}\sin(\pi\bar{\tau}) \\
&\quad - (2\pi)^2 b_{2\eta}\sin(2\pi\bar{\tau}) \qquad (53)\\
\tau_f^3\eta'''_\tau &= 6a_{3\eta} + 24a_{4\eta}\bar{\tau} + 60a_{5\eta}\bar{\tau}^2 - \pi^3 b_{1\eta}\cos(\pi\bar{\tau}) \\
&\quad - (2\pi)^3 b_{2\eta}\cos(2\pi\bar{\tau}). \qquad (54)
\end{aligned}$$

Equating these derivatives to the corresponding BCs, (5)–(10) yields a system of linear algebraic equations to solve for coefficients $a_{i\eta}$ and $b_{i\eta}$ ($\eta = \{x, y, z\}$). For instance, for the x-coordinate, the set of linear algebraic equations becomes

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & \pi & 2\pi \\
0 & 1 & 2 & 3 & 4 & 5 & -\pi & 2\pi \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 6 & 12 & 20 & 0 & 0 \\
0 & 0 & 0 & 6 & 0 & 0 & -\pi^3 & -8\pi^2 \\
0 & 0 & 0 & 6 & 24 & 60 & \pi^3 & -8\pi^2
\end{bmatrix}
\begin{bmatrix} a_{0x} \\ a_{1x} \\ a_{2x} \\ a_{3x} \\ a_{4x} \\ a_{5x} \\ b_{1x} \\ b_{2x} \end{bmatrix}$$

$$= \begin{bmatrix} x_0 \\ x_f \\ x'_0\tau_f \\ x'_f\tau_f \\ x''_0\tau_f^2 \\ x''_f\tau_f^2 \\ x'''_0\tau_f^3 \\ x'''_f\tau_f^3 \end{bmatrix}. \tag{55}$$

As seen, now that the order of parameterization has been increased, the third-order derivatives (jerks) at the initial and final points can be satisfied. In our case, since the jerks x'''_0 and x'''_f are not fixed, we can use them as two additional varied parameters alongside with τ_f.

Consequently, by resolving the system (55) the coefficients of the basis function can be obtained as follows (the coefficients for y and z coordinates have a similar form)

$$\begin{aligned}
a_{0x} &= x_0, \\
a_{1x} &= \frac{\left(1, \tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} -120(12x_0 - x_f) + \pi^2(x_0 - x_f) \\ -12\pi^2(9x'_0 - x'_f) + 8\pi^4 x'_0 \\ -72(7x''_0 + 3x''_f) + 12\pi^2(3x''_0 + x''_f) \\ -24(3x'''_0 - 2x'''_f) + \pi^2(5x'''_0 + 3x'''_f) \end{pmatrix}}{8(15 - \pi^2)(12 - \pi^2)} \\
a_{2x} &= \frac{x''_0\tau_f^2}{2}, \\
a_{3x} &= \frac{\left(1, \tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} 40(12\pi^2(x_0 - x_f) - \pi^4(x_0 - x_f)) \\ 4(15\pi^2(5x'_0 + 3x'_f) - 2\pi^4(3x'_0 + 2x'_f)) \\ 3\pi^2(26x''_0 - 5x''_f) - 2\pi^4(3x''_0 - x''_f) \\ 120x'''_0 - \pi^2(9x'''_0 + x'''_f) \end{pmatrix}}{4(15 - \pi^2)(12 - \pi^2)} \\
a_{4x} &= \frac{\left(1, \tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} -120\pi^2(12 - \pi^2)(x_0 - x_f) \\ -60\pi^2(13x'_0 + 10x'_f) - 8\pi^4(8x'_0 + 7x'_f) \\ 360(x''_0 - x''_f) + 60\pi^2(2x''_0 - x''_f) - 4\pi^4(3x''_0 - 2x''_f) \\ 120(2x'''_0 + x'''_f) - \pi^2(19x'''_0 + 11x'''_f) \end{pmatrix}}{8(15 - \pi^2)(12 - \pi^2)} \\
a_{5x} &= \frac{\left(1, \tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} 24\pi^2(x_0 - x_f) \\ 12\pi^2(x'_0 + x'_f) \\ 2(3 + \pi^2)(x''_0 - x''_f) \\ 3(x'''_0 + x'''_f) \end{pmatrix}}{4(15 - \pi^2)} \\
b_{1x} &= \frac{\left(\tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} 12(x'_0 - x'_f) \\ 6(x''_0 + x''_f) \\ x'''_0 - x'''_f \end{pmatrix}}{2\pi(12 - \pi^2)}, \\
b_{2x} &= \frac{\left(1, \tau_f, \tau_f^2, \tau_f^3\right)\begin{pmatrix} 120(x_0 - x_f) \\ 60(x'_0 + x'_f) \\ 12(x''_0 - x''_f) \\ x'''_0 - x'''_f \end{pmatrix}}{16\pi(15 - \pi^2)}.
\end{aligned} \tag{56}$$

Now, the total number of varied parameters increases to seven.

It should be noted that the derivatives in Eqs. (55) and (56) are defined in the virtual domain, while in fact they are given in the physical domain (see Eqs. (7)–(10)). The mapping is conducted in accordance with Eq. (45). By design, the speed factor $\lambda(\tau)$ simply scales the entire problem (the higher speed factor $\lambda(\tau)$, the larger τ_f) [16], so it may be assumed that

$$\lambda_{0;f} = 1, \quad \lambda'_{0;f} = 0, \ldots \tag{57}$$

which results in

$$\eta'_{\tau;0;f} = \dot{\eta}_{0;f}, \quad \eta''_{\tau\tau;0;f} = \ddot{\eta}_{0;f}, \ldots. \tag{58}$$

4.3. Optimization routine

Now let us describe the numerical procedure for finding the optimal solution among all candidate trajectories given by Eq. (51). To assure Eq. (57) the initial guess on the length of the virtual arc τ_f is set to be proportional to the physical distance between the starting and terminal points

$$\tau_f = 1.5\sqrt{(x_f - x_0)^2 + (y_f - y_0)^2 + (z_f - z_0)^2}. \tag{59}$$

With this guess and also guessing on the values of other varied parameters, $x'''_0, x'''_f, y'''_0, y'''_f, z'''_0$, and z'''_f, the coefficients of candidate trajectory are computed using Eq. (56).

Having an analytical representation of candidate trajectory the values of x_j, y_j, z_j, and x_j', y_j', z_j', $j = 1, \ldots, N$ are computed over the set of N nodes, evenly spaced along the virtual arc $[0; \tau_f]$

$$\begin{aligned}
&\tau_j = \tau_{j-1} + \Delta\tau \quad (\bar{\tau}_j = \bar{\tau}_{j-1} + \Delta\bar{\tau}), \quad j = 2, \ldots, N, \\
&(\tau_1 = \bar{\tau}_1 = 0)
\end{aligned} \tag{60}$$

where

$$\Delta\tau = \tau_f(N-1)^{-1}, \quad (\Delta\bar{\tau} = (N-1)^{-1}). \tag{61}$$

The time step is calculated based on the distance between the two computational nodes along the arc and speed

$$\Delta t_{j-1} = \frac{\sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2}}{\sqrt{u_{j-1}^2 + w_{j-1}^2 + c_x^2 + c_y^2 + 2c_x(u_{j-1}\cos(\psi_{j-1})) + 2c_y(u_{j-1}\sin(\psi_{j-1}))}} \tag{62}$$

which consequently produces

$$\lambda_j = \Delta\tau\,\Delta t_{j-1}^{-1}. \tag{63}$$

The rest of states and controls are computed in accordance with Eqs. (46)–(48). The derivatives $u'_{\tau;j}$, $w'_{\tau;j}$ and $r'_{\tau;j}$ can be computed using finite differences or analytical expressions. For example, the forward and vertical velocity vector components can be computed as follows

$$u'_{\tau;j} = \frac{\left((\lambda'_j x'_{\tau;j} + \lambda_j x''_{\tau;j})(\lambda_j x'_{\tau;j} - c_x) + (\lambda'_j y'_{\tau;j} + \lambda_j y''_{\tau;j})(\lambda_j y'_{\tau;j} - c_y)\right)}{u_j} \tag{64}$$

$$w'_{\tau;j} = \lambda'_j z'_{\tau;j} + \lambda_j z''_{\tau;j}.$$

When all parameters (states and controls) are computed in each of the N points, the performance index is computed as follows:

$$J = k_T\left(\sum_{j=1}^{N-1}\Delta t_j - T\right)^2 + k_u\sum_{j=1}^{N-1}(T_{u;j}^2 + T_{w;j}^2 + T_{r;j}^2)\Delta t_j. \tag{65}$$

The penalty function that accounts for violation of constraints on the states and controls is constructed as

$$\begin{aligned}\Delta = {} & k_\psi\left(\psi_N - \psi_f\right)^2 + k_r\max_j\left(0; \left|r_j\right| - r^{\max}\right)^2 \\ & + k_{T_u}\max_j\left(0; \left|T_{u;j}\right| - T_u^{\max}\right)^2 \\ & + k_{T_w}\max_j\left(0; \left|T_{w;j}\right| - T_w^{\max}\right)^2 + k_{T_r}\max_j\left(0; \left|T_{r;j}\right| - T_r^{\max}\right)^2.\end{aligned} \tag{66}$$

In Eqs. (65) and (66) k_i are the weighting coefficients allowing all individual terms to be balanced. As mentioned in Section 3.3 for consistency, the MATLAB *fmincon* function was used with IDVD as well. The results of simulations are presented in the next section.

5. IDVD simulations results

In this section, performance of the proposed IDVD-guidance method is demonstrated in the controlled and then realistic environment. The controlled environment simulations were conducted to demonstrate a superb performance of the IDVD method and its explicit readiness to be implemented on board of a real AUV. They also included assessment of algorithm robustness with respect to varying BCs. Realistic environment simulations included cluttered operational environment and uncertainties in the DS position.

5.1. Solution feasibility and computational performance

First, the developed IDVD-method-based approach was tested in the same no-fly-zone "static" scenario as used in simulations presented in Section 3.3. Figs. 11–13 demonstrate the results of computer simulation in this case. For comparison, the 7- and 10-state LGLPS solutions are also added. As seen from Fig. 11 while the LGLPS singular arc solution results in essentially straight dive from the starting point to the final DS position, the IDVD trajectory is more curved as dictated by the chosen parametrizations. Figs. 12 and 13 shows the time histories of the vehicle's forward and vertical velocities along with its yaw angle and yaw rate indicating satisfaction of all BCs, and particularly a good alignment of the vehicle with the DS at the terminal point. The IDVD (and LGLPS) generated controls are depicted in Fig. 13.

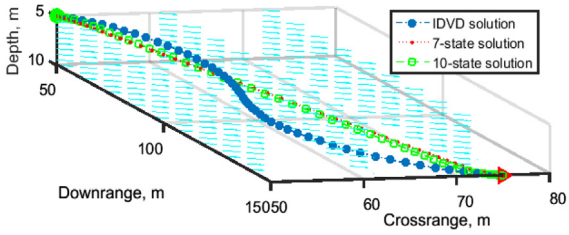

Fig. 11. The 3D path for traveling from the initial to DS position.

Compared to the LGLPS method, parameterizations (51) used by the IDVD-method formulation in this particular case do not allow a singular arc control. As the first-order optimality conditions are not enforced in the IDVD structure, the trajectory cannot be called optimal in the classical OCP sense. The IDVD method does, however, try to find the best trajectory among all candidate trajectories within the chosen parameterizations, and it certainly produces a feasible trajectory featuring smooth transitions at the departure and arrival points (Fig. 14). By design the IDVD-computed controls are the smoothest ones accounting for the actual dynamics and therefore readily realizable on board. As seen from Fig. 13, while the specific bounds established for the 10-state LGLPS solution, $\upsilon_u^{\max}$, $\upsilon_w^{\max}$ and $\upsilon_r^{\max}$, do make the departure and arrival conditions smoother, they do not necessarily cause a complete match with the IDVD solution (except the initial part in the surge speed control). It should be noted that introducing more strict bounds in the heave speed and yaw rate control would cause a better match, but that was not an objective of this study.

Fig. 15 shows arc histories of speed factor and its derivatives resulting in a non-linear mapping between the virtual and physical domains.

Even though the performance index for the IDVD solution is about 24% higher than for the "optimal" 10-state solution obtained using the LGLPS method (Fig. 16), the IDVD-based approach provides a huge computational advantage (Fig. 17). The required central processing unit (CPU) time is also shown in Table 1. For example, comparing the 50-node solutions, the IDVD solution converges 47.1 times faster than that of 10-state LGLPS (as shown by the figures in parentheses in Table 1). Also, increasing the number of nodes in the LGLPS method means increasing the number of varied parameters and as a result — drastically increases the required CPU time. Increasing the number of computational nodes in the IDVD method does not affect the number of varied parameters. In addition, while the IDVD approach converges to some feasible trajectory every time, the convergence of LGLPS approach is not guaranteed. As seen from Fig. 16 an attempt to make LGLPS solution more feasible by increasing the number of states inevitably leads to increase of the performance index value. As mentioned above, tightening $\upsilon_u^{\max}$, $\upsilon_w^{\max}$ and $\upsilon_r^{\max}$, would make the performance index values of LGLPS and IDVD methods even closer.

5.2. Robustness of the IDVD solution in SITL simulations

Fig. 18 shows the locations of the LGL and IDVD nodes for different number of nodes involved in Figs. 16—17. By design, they are not spaced evenly along the time axis and that is why on-board implementation includes the Interpolator (Fig. 3). As shown in Fig. 3, a piecewise cubic Hermite interpolating polynomial is used

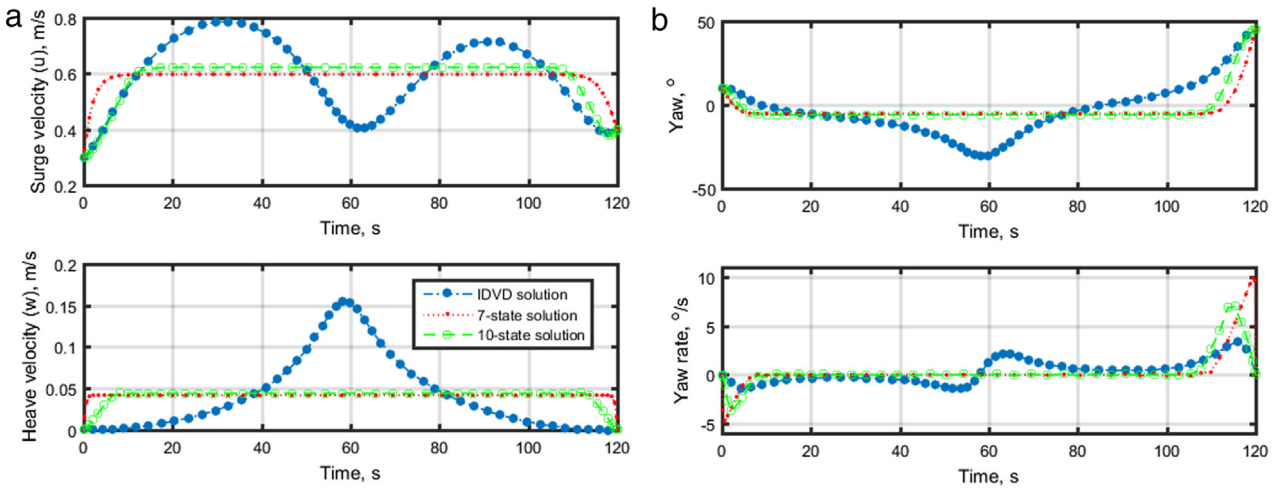

Fig. 12. Time history of surge and heave velocities (a), along with vehicle's yaw angle and yaw rate (b).

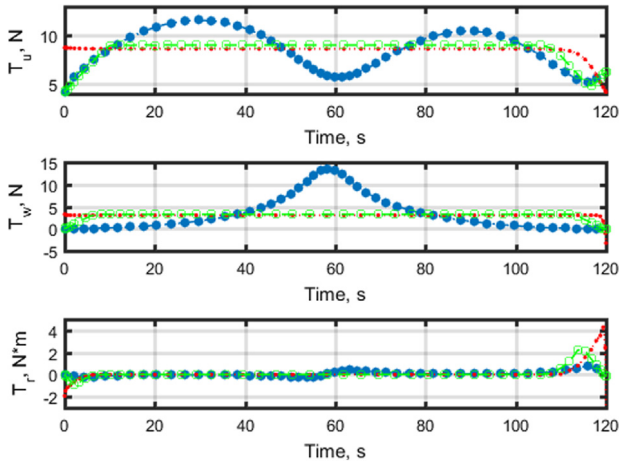

Fig. 13. Control time histories.

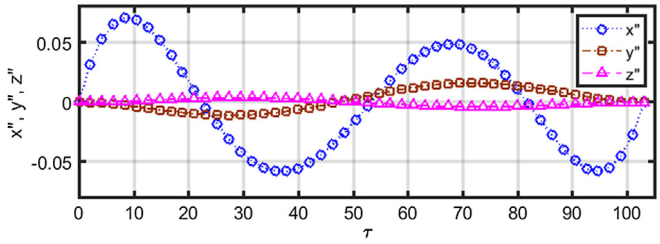

Fig. 14. Second-order derivatives of AUV's coordinates.

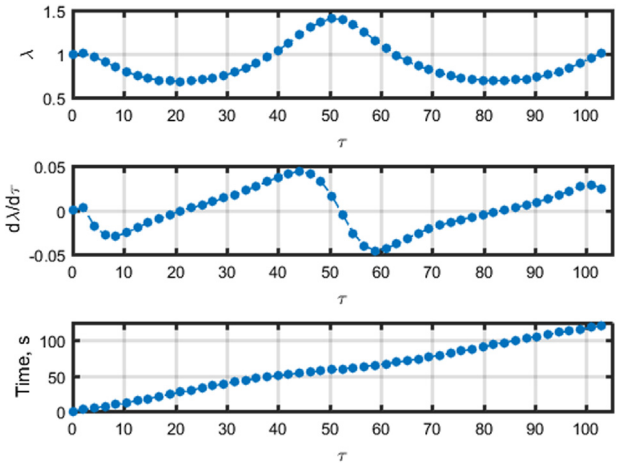

Fig. 15. Arc histories of IDVD's speed factor and mapping between the virtual and physical domains.

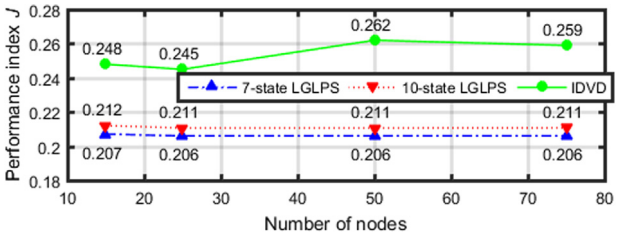

Fig. 16. Performance index for IDVD and LGLPS solutions using the different number of the computational nodes.

to interpolate the reference trajectory states at $M > N$ evenly-spaced nodes.

To examine the quality of interpolated solution, the following test was conducted. The reference trajectory controls were also interpolated at the same M evenly-spaced nodes and then used to integrate ordinary differential equations (1)–(4) from $t = 0$ to $t = T$ starting from the initial conditions (5). The objective was to see whether integrated trajectory happened to be close to the interpolated one. Two metrics referred to as the final position and final heading errors, $\Delta\rho$ and $\Delta\psi$, respectively, were defined by the DS geometry (see Fig. 2) as follows:

$$\Delta\rho = \sqrt{(x(t_f) - x_f)^2 + (y(t_f) - y_f)^2 + (z(t_f) - z_f)^2} \leq 0.6 \text{ m}$$
$$\Delta\psi = \left|\psi(t_f) - \psi_f\right| \leq 9^\circ. \quad (67)$$

In Eq. (67), $x(t_f)$, $y(t_f)$, $z(t_f)$, and $\psi(t_f)$ denote the final values of the integrated solution. The position error tolerance is based on the distance of the vehicle's center of gravity from the DS position (the center of the outer end of the DS cone). Specifically, for the DS described in [26], the position error in Eq. (67) is defined as a radius of a sphere that has the same cross-section as the outer end of the DS cone [27]. For heading error tolerance is defined by the DS cone entrance angle [27].

Table 1
The CPU time for the IDVD and LGLPS methods (numbers in parenthesis show time relative to IDVD time).

Method	15-node solution	25-node solution	50-node solution	75-node solution
IDVD	3	4	7	7
7-state LGLPS	4 (1.3)	16 (4.0)	94 (13.4)	281 (40.1)
10-state LGLPS	7 (2.3)	39 (9.8)	330 (47.1)	1016 (145.1)

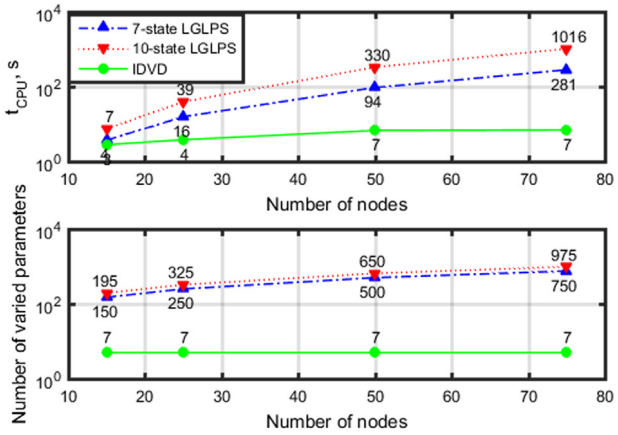

Fig. 17. Computational effectiveness of the IDVD and LGLPS solutions.

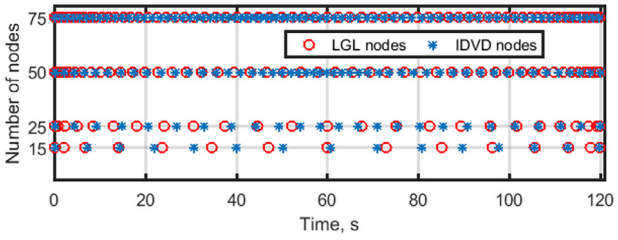

Fig. 18. Distribution of the LGL and IDVD nodes along the time interval.

Specifically, the standard 4th-order Runge–Kutta method with a fixed time step of 0.01 s was used as a propagator (in this case $M = 12\,000$), and the achieved results, dependent on the number of nodes, are presented in Table 2.

As seen from this table, compared to the IDVD solution itself, which assures $\Delta\rho = 0$ and $\Delta\psi = 0$ by construction, the propagated solution (based on integrating the controls) indeed leads to some errors. These errors are of the order of the tolerances in Eq. (67) and decrease with increasing the number of nodes that was used for obtaining the controls time histories for the IDVD solution. These errors happen to be small enough to be corrected by the AUV controller, thus enabling the usage of the interpolated solutions for the state vector by the Controller block (Fig. 3). Moreover, if the control architecture would include the feedforward loop, the interpolated controls could then be used as well.

The ultimate robustness test involved the complete SITL simulation environment including the actual controller as shown in Fig. 3. In this test the interpolated desired states were passed on to SMC, which computed the location of the 1-meter-ahead traveling WP and formed an input vector for the SMC block to produce the actual controls.

The SITL simulation results for the TPBV problem solution discussed in the previous section are shown in Figs. 19, 20. As seen, despite the intentionally introduced discrepancies in controls at the beginning of trajectory and discrepancies in the model used in optimization (Eqs. (1)–(4)) and a full-degree of freedom model of the AUV used within the SITL simulation environment, the controller

Table 2
Arrival accuracy performance of the solution obtained by integrating the IDVD-provided controls.

Number of nodes	$\Delta\rho$, m	$\Delta\psi$, °
15	1.60	3.80
25	0.90	0.88
50	0.86	0.16
75	0.38	0.41

Table 3
Arrival accuracy performance of the 50-node solutions in SITL simulations.

Method	$\Delta\rho$, m	$\Delta\psi$, °
IDVD	0.1	1.1
7-state LGLPS	0.2	8.2
10-state LGLPS	0.7	0.5

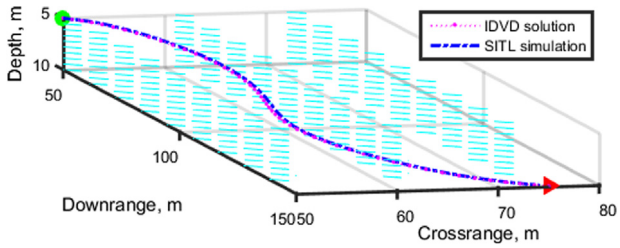

Fig. 19. Optimized and propagated IDVD trajectories.

does a good job of tracking the interpolated reference trajectory. A peculiarity of the SMC implementation with the Traveling WP block causes about 2-second lead time (1 m distance divided by the average speed of about 0.5 m/s) as compared to the interpolated reference trajectory. Hence, to accommodate the traveling WP beyond the final point, the final approach leg is extended along the DS centerline. That allows terminating simulation at exactly 120 s.

Since in this particular scenario the reference trajectory can be generated ahead of time (there is no limit on t_{CPU}), Table 3 compares the results of using the SITL simulation environment of Fig. 3 with the reference trajectories presented and discussed in Sections 3.3 and 4.1. As expected, due to the relatively large accelerations required at the end of the trajectory when using LGPS solutions compared to the IDVD solution, the final position and final heading errors in tracking the LGPS solutions are greater than those for the IDVD solution.

Finally, to assess the performance of the IDVD-based guidance method while varying BCs the Monte Carlo trials were executed. In these trials The BCs were sampled using the uniform distribution $U(\ell^b, u^b)$, with ℓ^b and u^b being the lower and upper bounds, respectively. In the first 200 trials initial conditions for x, y, z, and ψ were randomly chosen from the hypercube defined by $x_0 \pm 5$ m, $y_0 \pm 5$ m, $z_0 \pm 2$ m, and $\psi_0 \pm 20°$, where the nominal values were defined in Eq. (37). For the second set of 200 trials final conditions were varied within $x_f \pm 5$ m, $y_f \pm 5$ m, $z_f \pm 2$ m, and $\psi_f \pm 20°$. In addition to that, in both sets of trials there was a 10% variation (uncertainty) applied to hydrodynamic coefficients used within the SITL simulation platform.

The Monte Carlo trials were realized in MATLAB using a parallel implementation on a high performance cluster machine that has

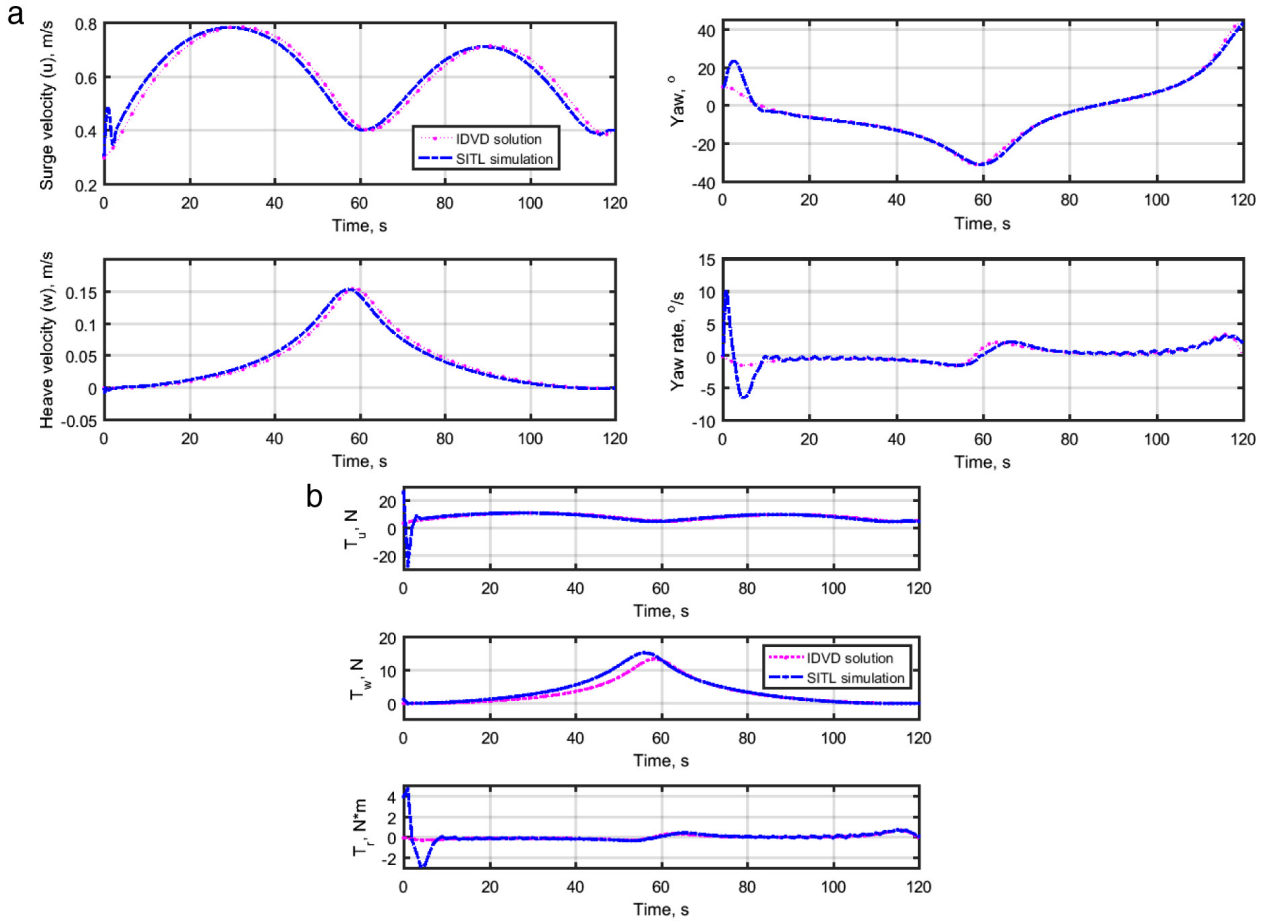

Fig. 20. Control time histories of the optimized and propagated IDVD solutions.

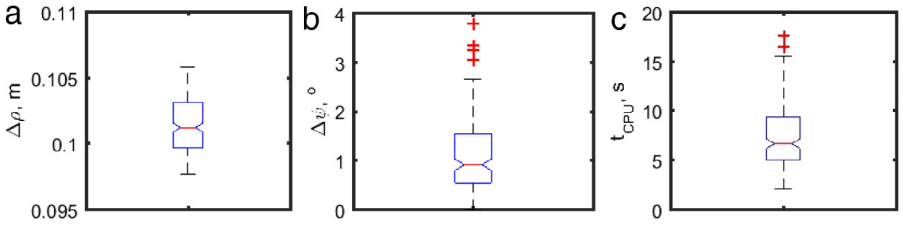

Fig. 21. Distributions of the final position (a) and final heading errors (b), and computational effectiveness (c) while executing Monte Carlo simulations with varying the terminal conditions.

1160 CPU cores and 4.25 TB RAM. All 400 runs satisfied Eq. (67) proving acceptable robustness of the developed algorithm. As an example, Figs. 21(a), (b) shows the box plots of the final position and final heading errors while varying the terminal conditions. Fig. 21(c) proves the computational effectiveness of developed algorithm.

5.3. Performance with a realistic docking scenario

So far, all examples addressed a simplified TPBV problem with no obstacles (15) and no uncertainties. In practice, these real-world features should obviously be included in simulation, and performance of the guidance method be tested. To this end, this section shows the results of a simulation in the case when there is an uncertainty in the actual location of DS. In this scenario, it is assumed that the AUV executing a docking mission gets an update about DS pose every t_{sample} using an ultra-short-baseline (USBL) sensor. The DS pose information is corrupted by both sensor and environmental noises. As the AUV approaches the DS, the impact of uncertainty reduces, and the DS pose information becomes more accurate. The USBL acoustic positioning system used in simulations (denoted as SA Sensors in Fig. 3) is adapted from [42]. Moreover, to make the scenario more challenging and assess the IDVD-method based guidance performance in a cluttered environment six no-fly zones modeled in Eq. (15) were added to the operating field.

In this scenario, the initial conditions together with the constraints over the AUV's states and controls, are set as per the previous scenario described in Section 3. As opposed to the fixed values in Eq. (37), the first three elements of the final state vector are dependent on the range from the true DS position, D, and modeled as

$$\begin{aligned} x_f(D) &= 180(1+\delta(D))\,\mathrm{m} \\ y_f(D) &= 70(1+\delta(D))\,\mathrm{m} \\ z_f(D) &= 11(1+\delta(D))\,\mathrm{m} \end{aligned} \tag{68}$$

Online trajectory planning mechanism

- Set $\mathbf{x}_0$ and $\mathbf{x}_f$ for the candidate trajectories
- Set the cost function Eq.16 for the candidate trajectories
- Generate the first set of candidate trajectories for the docking operation
- Check Replanning Flag continuously

BEGIN

If *Replanning Flag ==1*

- Apply the current states ($\mathbf{x}(t)$) and new final docking condition ($\mathbf{x}_{DS}$) as follows:
- $\mathbf{x}_0 \leftarrow \mathbf{x}(t)$
- $\mathbf{x}_f \leftarrow \mathbf{x}_{DS}$ (provided that a new ping from USBL is available)
- Use the solution at hand (from previous trajectory) as an initial solution or initial guess for the IDVD optimization variables
- Generate a new set of candidate trajectories
- Check the problem feasibility/optimality conditions

If the conditions are satisfied
Follow the new solution
Terminate the planning process
Otherwise
Follow the pervious solution
Terminate the planning process
Back to **BEGIN**
End
End
END

Fig. 22. Trajectory re-optimization routine.

where $\delta(D) = N(0, \sigma(D)^2)$ represents a normally distributed uncertainty in knowing the DS position with the standard deviation $\sigma(D) = 4.1 \cdot 10^{-6} D^2$. As a result, at each DS position update, every t_{sample}, the AUV reference trajectory needs to be recomputed to account for an updated $\mathbf{x}_f(D(t))$. When doing so, the current AUV state is used as a new initial state

$$\mathbf{x}_0 = \mathbf{x}(t) \tag{69}$$

(that is where a capability to reinforce Eq. (11) becomes essential). Fig. 22 shows the flowchart of the utilized online (on-the-go) trajectory re-optimization procedure.

Figs. 23–25 demonstrate an example where a reference trajectory was updated twice based on two sequential USBL updates on the DS position. Fig. 23, illustrates the generated path in 3D, revealing no-fly zones and the same current field as in the offline computation scenario of Section 3.3. Three solid circles in Fig. 23 (including the starting point) indicate the DS position update points and three triangles show the corresponding perception of the DS position. Specifically, the Trajectory generator block (see Fig. 3) generates the first reference trajectory based on the DS position information available at the first solid circle, denoted as *Start*. At this point, the DS position is thought to be at the location denoted as *1st destination*. With this reference trajectory, the AUV continues its motion to this destination. At the first update corresponding to the AUV position depicted by the second solid circle (*1st update*), the vehicle receives a new ping from the USBL and based on the new information, the Trajectory generator block refines the reference trajectory and generates a new one leading to the *2nt destination* point. The AUV keeps tracking the second reference trajectory until the next ping from the DS is received, and the 2nd update occurs. Another reference trajectory is then generated with respect to the updated estimate of the DS position denoted as the *Actual DS* in Fig. 23 (at this point, being about 40 m away from the DS, its horizontal and vertical position is known to within 1 m and 0.1 m, respectively). As seen in Fig. 23, each of the three reference trajectories forces the AUV to maneuver around no-fly zones and that is where IDVD-approach capability to generate spatial non-singular arc solutions really pays off.

Fig. 24(a)-(b) feature time histories of surge and heave velocities and yaw angle and yaw rate, corresponding to all three reference trajectories shown in Fig. 23, respectively. Fig. 24(b) clearly shows a correct final zero-yaw-rate aligning the AUV with the DS centerline.

Data shown in Fig. 25 proves that all controls are within their limits (within preset tolerances). In the particular realization shown in Fig. 23, where the new DS position updates happen to be farther and farther away than originally expected, obtaining the new reference trajectories becomes more and more challenging. In this particular simulation, the only way to be able to arrive at the DS in exactly 120 s with zero acceleration was to allow the final surge velocity to vary. As seen from Fig. 25, the surge velocity control reached its limit at the second update and was within 1% above its limit for the third update. The performance index values obtained for the three updates are also indicative of this. While $J_{update\ 1} = 0.26$, it steadily increases for the following updates, $J_{update\ 2} = 0.68$ and $J_{update\ 3} = 0.85$, respectively.

6. Conclusion

The goal of this paper was to present a real-time trajectory generator algorithm for underwater docking operations and verify its performance using the SITL simulation environment created for an AUV, which is currently in the development phase. The high-fidelity AUV model adopted in this study was chosen in an attempt to accommodate the AUV's dynamics at the stage of the reference trajectory generation and thus make it more feasible and readily available to be passed on to the AUV's controller for tracking without saturating the AUV's thrusters. Since the actual trajectory needs to be very close to that generated by the Trajectory generator, it is very important to assure high accuracy in satisfying the boundary conditions and to avoid the no-fly zones in the cluttered operational environment. The solutions obtained using the IDVD-method-based algorithm were analyzed from the standpoint of their feasibility, optimality and computational effectiveness. The developed algorithm was thoroughly tested within the SITL simulation environment and proved its robustness with respect to unmodeled dynamics, BC variations, and uncertainties associated with the perceived position of the DS. Its computation effectiveness allows the reference trajectory to be updated in real

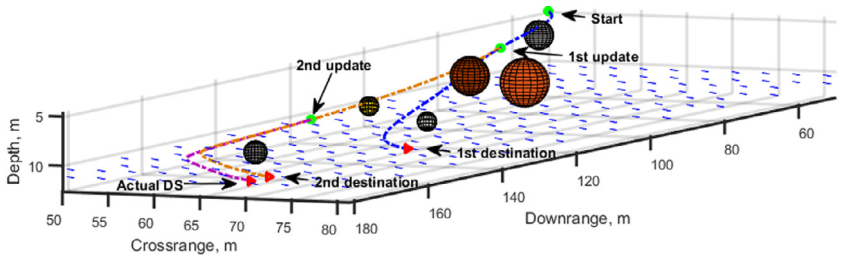

Fig. 23. The 3D collision-free trajectory re-optimized based on a better knowledge of the DS location.

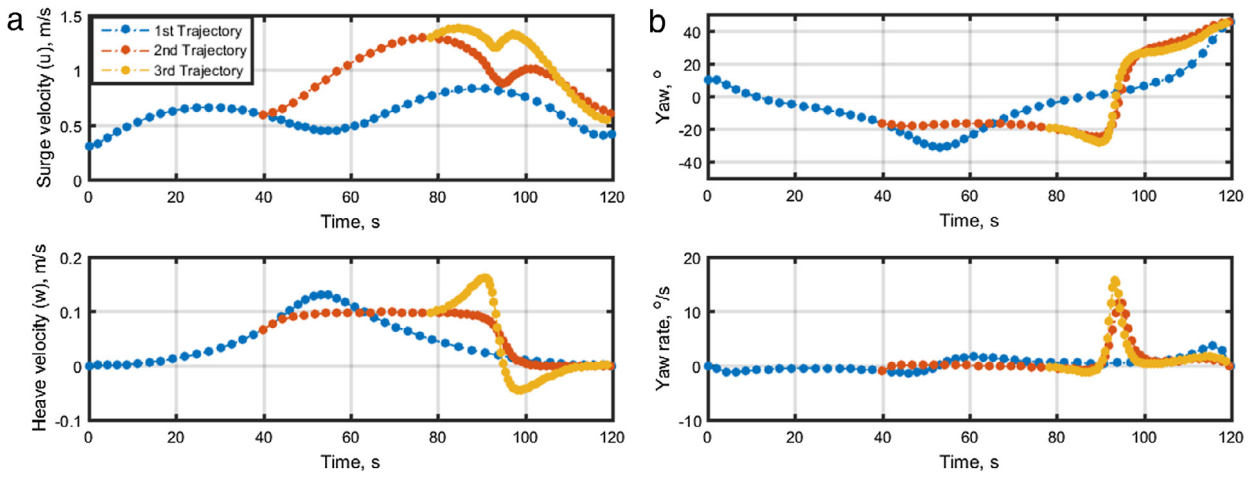

Fig. 24. Refinement of surge and heave velocities (a), along with the yaw angle and yaw rate updates (b).

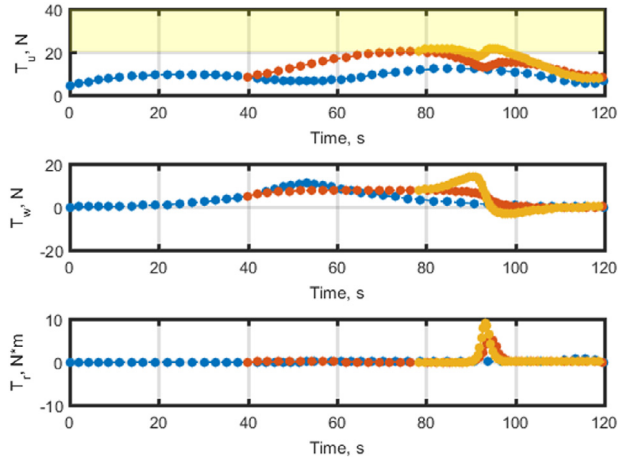

Fig. 25. Time histories of the control inputs for online docking trajectory generation.

time as needed, and therefore makes it a good candidate for future testing on a real AUV.

References

[1] B. Allen, T. Austin, N. Forrester, R. Goldsborough, A. Kukulya, G. Packard, M. Purcell, R. Stokey, Autonomous docking demonstrations with enhanced REMUS technology, MTS/IEEE Conference OCEANS'06, 2006, pp. 1-6. http://dx.doi.org/10.1109/OCEANS.2006.306952.

[2] M.D. Feezor, F.Y. Sorrell, P.R. Blankinship, J.G. Bellingham, Autonomous underwater vehicle homing/docking via electromagnetic guidance, IEEE J. Ocean. Eng. 26 (4) (2001) 515–521. http://dx.doi.org/10.1109/48.972086.

[3] D.-J. Li, Y.-H. Chen, J.-G. Shi, C.-J. Yang, Autonomous underwater vehicle docking system for cabled ocean observatory network, Ocean Eng. 109 (2015) 127–134. http://dx.doi.org/10.1016/j.oceaneng.2015.08.029.

[4] O.A. Yakimenko, S.P. Kragelund, Real-time optimal guidance and obstacle avoidance for UMVs, Autonomous Underwater Vehicles, InTech (2011). http://dx.doi.org/10.5772/23401.

[5] A.M. Yazdani, K. Sammut, A. Lammas, Y. Tang, Real-time quasi-optimal trajectory planning for autonomous underwater docking, IEEE International Symposium on Robotics and Intelligent Sensors, IRIS, 2015, pp. 15-20. http://dx.doi.org/10.1109/IRIS.2015.7451579.

[6] B.M. Ferreira, A.C. Matos, N.A. Cruz, A.P. Moreira, Homing a robot with range-only measurements under unknown drifts, Robot. Auton. Syst. 67 (C) (2015) 3–13. http://dx.doi.org/10.1016/j.robot.2014.09.035.

[7] Y. Li, Y. Jiang, J. Cao, B. Wang, Y. Li, AUV docking experiments based on vision positioning using two cameras, Ocean Eng. 110 (A) (2015) 163–173. http://dx.doi.org/10.1016/j.oceaneng.2015.10.015.

[8] W. Naeem, R. Sutton, S.M. Ahmad, Pure pursuit guidance and model predictive control of an autonomous underwater vehicle for cable/pipeline tracking, J. Mar. Sci. Environ. C (1) (2003) 25–35.

[9] J.-Y. Park, B.-H. Jun, P.-M. Lee, Y.-K. Lim, J.-H. Oh, Modified linear terminal guidance for docking and a time-varying ocean current observer, IEEE Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2011, pp. 1-6. http://dx.doi.org/10.1109/UT.2011.5774141.

[10] K. Teo, B. Goh, O.K. Chai, Fuzzy docking guidance using augmented navigation system on an AUV, IEEE J. Ocean. Eng. 40 (2) (2015) 349–361. http://dx.doi.org/10.1109/JOE.2014.2312593.

[11] I. Spangelo, O. Egeland, Trajectory planning and collision avoidance for underwater vehicles using optimal control, IEEE J. Ocean. Eng. 19 (4) (1994) 502–511. http://dx.doi.org/10.1109/48.338386.

[12] M. Chyba, T. Haberkorn, R.N. Smith, S.K. Choi, Design and implementation of time efficient trajectories for autonomous underwater vehicles, Ocean Eng. 35 (1) (2008) 63–76. http://dx.doi.org/10.1016/j.oceaneng.2007.07.007.

[13] M. Chyba, T. Haberkorn, S.B. Singh, R.N. Smith, S.K. Choi, Increasing underwater vehicle autonomy by reducing energy consumption, Ocean Eng. 36 (1) (2009) 62–73. http://dx.doi.org/10.1016/j.oceaneng.2008.07.012.

[14] R.P. Kumar, A. Dasgupta, C. Kumar, Real-time optimal motion planning for autonomous underwater vehicles, Ocean Eng. 32 (11–12) (2005) 1431–1447. http://dx.doi.org/10.1016/j.oceaneng.2004.11.010.

[15] J.T. Betts, Survey of numerical methods for trajectory optimization, J. Guid. Control Dyn. 21 (2) (1998) 193–207. http://dx.doi.org/10.2514/2.4231.

[16] O.A. Yakimenko, Direct method for rapid prototyping of near-optimal aircraft trajectories, J. Guid. Control Dyn. 23 (5) (2000) 865–875. http://dx.doi.org/10.2514/2.4616.

[17] R. Jamilnia, A. Naghash, Optimal guidance based on receding horizon control and online trajectory optimization, J. Aerosp. Eng. 26 (4) (2011) 786–793. http://dx.doi.org/10.1061/(ASCE)AS.1943-5525.0000194.

[18] Y. Zhang, J. Chen, L. Shen, Real-time trajectory planning for UCAV air-to-surface attack using inverse dynamics optimization method and receding horizon control, Chin. J. Aeronaut. 26 (4) (2013) 1038–1056. http://dx.doi.org/10.1016/j.cja.2013.04.040.

[19] I. Kaminer, O. Yakimenko, V. Dobrokhodov, A. Pascoal, N. Hovakimyan, V. Patel, C. Cao, A. Young, Coordinated path following for time-critical missions of multiple uavs via L1 adaptive output feedback controllers, AIAA Guidance, Navigation and Control Conference and Exhibit, 2007. http://dx.doi.org/10.2514/6.2007-6409.

[20] J. Lukacs, O. Yakimenko, Trajectory-shaping guidance for interception of ballistic missiles during the boost phase, J. Guid. Control Dyn. 31 (5) (2008) 1524–1531. http://dx.doi.org/10.2514/1.32262.

[21] I. Cowling, O. Yakimenko, J. Whidborne, A. Cooke, Direct method based control system for an autonomous quadrotor, J. Intell. Robot. Syst. 60 (2) (2010) 285–316. http://dx.doi.org/10.1007/s10846-010-9416-9.

[22] N. Slegers, O. Yakimenko, Terminal guidance of autonomous parafoils in high wind to airspeed ratios, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 225 (3) (2011) 336–346. http://dx.doi.org/10.1243/09544100JAERO749.

[23] G. Boyarko, M. Romano, O. Yakimenko, Time-optimal reorientation of a spacecraft using an inverse dynamics optimization method, J. Guid. Control Dyn. 34 (4) (2011) 1197–1208. http://dx.doi.org/10.2514/1.49449.

[24] J. Ventura, M. Romano, U. Walter, Performance evaluation of the inverse dynamics method for optimal spacecraft reorientation, Acta Astronaut. 110 (2015) 266–278. http://dx.doi.org/10.1016/j.actaastro.2014.11.041.

[25] I.M. Ross, F. Fahroo, Legendre pseudospectral approximations of optimal control problems, in: Lecture Notes in Control and Inforrmation Sciences, 295, 2003, pp. 327–342.
[26] J.-Y. Park, Docking Problem and Guidance Laws Considering Drift for an Underactuated Autonomous AUV, (Ph.D. dissertation), Korea Advanced Institute of Science and Technology, 2011.
[27] A. Lammas, F. He, K. Sammut, 6-DoF navigation systems for autonomous underwater vehicles, Mobile Robots Navigation, InTech (2010). http://dx.doi.org/10.5772/8978.
[28] M. Kokegei, F. He, K. Sammut, Fully coupled 6 degree-of-freedom control of an over-actuated autonomous underwater vehicle, Autonomous Underwater Vehicles, InTech (2011). http://dx.doi.org/10.5772/25064.
[29] Z. Zeng, A. Lammas, K. Sammut, F. He, Y. Tang, Shell space decomposition based path planning for AUVs operating in a variable environment, Ocean Eng. 91 (2014) 181–195. http://dx.doi.org/10.1016/j.oceaneng.2014.09.001.
[30] T.I. Fossen, Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles, ISBN: 8292356002, 2002.
[31] E.D.B. Medagoda, P.W. Gibbens, Synthetic-waypoint guidance algorithm for following a desired flight trajectory, J. Guid. Control Dyn. 33 (2) (2010) 601–606. http://dx.doi.org/10.2514/1.46204.
[32] L.A. Gonzalez, Design Modelling and Control of an Autonomous Underwater Vehicle, (Bachelor of Engineering Honours thesis), The University of Western Australia, 2004.
[33] T.T.J. Prestero, Verification of a Six-Degree of Freedom Simulation Model for the REMUS Autonomous Underwater Vehicle, (Master of Science thesis), Massachusetts institute of technology, 2001.
[34] D.S. Naidu, Optimal Control Systems, CRC press, ISBN: 9781482292299, 2002.
[35] D. Garg, W.W. Hager, A.V. Rao, Pseudospectral methods for solving infinite-horizon optimal control problems, Automatica 47 (4) (2011) 829–837. http://dx.doi.org/10.1016/j.automatica.2011.01.085.
[36] F. Fahroo, I.M. Ross, Pseudospectral methods for infinite-horizon nonlinear optimal control problems, J. Guid. Control Dyn. 31 (4) (2008) 927–936. http://dx.doi.org/10.2514/1.33117.
[37] I.M. Ross, F. Fahroo, Pseudospectral knotting methods for solving nonsmooth optimal control problems, J. Guid. Control Dyn. 27 (2004) 397–405.
[38] D.R. Herber, Basic implementation of multiple-interval pseudospectral methods to solve optimal control problems, in: Technical Report UIUC-ESDL-2015-01, University of Illinois, 2015.
[39] F. Fahroo, I.M. Ross, Costate estimation by a Legendre pseudospectral method, J. Guid. Control Dyn. 24 (2) (2001) 270–277. http://dx.doi.org/10.2514/2.4709.
[40] O. Yakimenko, Y. Xu, G. Basset, Computing short-time aircraft maneuvers using direct methods, Int. J. Comput. Syst. Sci. 49 (3) (2010) 145–176. http://dx.doi.org/10.1134/S1064230710030159.
[41] D. Garg, M. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, G.T. Huntington, A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica 46 (11) (2010) 1843–1851. http://dx.doi.org/10.1016/j.automatica.2010.06.048.
[42] D. Pearson, E. An, M. Dhanak, K. von Ellenrieder, P. Beaujean, High-level fuzzy logic guidance system for an unmanned surface vehicle (USV) tasked to perform autonomous launch and recovery (ALR) of an autonomous underwater vehicle (AUV), IEEE/OES Autonomous Underwater Vehicles Conference, 2014. http://dx.doi.org/10.1109/AUV.2014.7054403.

Amir Mehdi Yazdani received his Master degree in Mechatronic and Automatic Control from Universiti Teknologi Malaysia in 2012. He is currently a Ph.D. student at the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia and has been the recipient of the FIPRS scholarship in 2013. His particular areas of research specialization are concerned with guidance of unmanned vehicles, optimal control and state estimation theory, and intelligent control applications.

Karl Sammut completed his Ph.D. at The University of Nottingham (U.K) in 1992 and was employed between 1992 and 1995 as a Postdoctoral Fellow with the Politecnico di Milano (Italy), and with Loughborough University (UK). He commenced his appointment at Flinders University in 1995. He is currently an Associate Professor with the Engineering Discipline in the School of Computer Science, Engineering and Mathematics. He is the Director of the Centre for Maritime Engineering, Control and Imaging at Flinders University. His areas of research specialization are concerned with navigation, optimal guidance and control systems, and mission planning systems for autonomous surface and underwater vehicles.

Oleg Yakimenko is a professor of systems engineering and a professor of mechanical and aerospace Engineering at the Naval Postgraduate School, Monterey, CA. He holds two doctoral degrees — in Aerospace Engineering (1991) and in Operations Research (1996). He serves as the Program Manager of the Autonomous Systems track, Director of the Aerodynamic Decelerator Systems Center, and Director of Autonomous Systems Engineering and Integration Laboratory. His areas of interest include modeling, guidance, navigation and control of manned and unmanned aerial and maritime vehicles, satellites, guided weapons and parachutes. Dr. Yakimenko is an author or co-author of almost 300 publications including 14 textbooks. He is Associate Fellow of the American Institute of Aeronautics and Astronautics and Fellow of the Russian Academy of Sciences of Aviation and Aeronautics.

Andrew Lammas received his Bachelor of Engineering (Computer Systems) from Flinders University, Adelaide, Australia in 2004. He completed his Ph.D. in Engineering also at Flinders University in 2012 focusing on underwater navigation. His research interests include design and implementation of model based robust nonlinear filtering methods for navigation and awareness systems within autonomous vehicles. He is currently a research associate within the Centre of Maritime Engineering, Control, and Imaging (CMECI) in the School of Computer Science, Engineering and Mathematics (CSEM), Flinders University of South Australia. In his current role he is working on the development of localization and mapping systems for automated inspection robots operating in confined spaces, as well as on battery pack modeling for large scale energy storage systems.

Youhong Tang is a senior lecturer and an Australian Research Council-Discovery Early Career Researcher (ARCDECRA) fellow at Flinders University. He is a research leader in the Centre for Maritime Engineering, Control and Imaging. He obtained his Ph.D. degree in the Hong Kong University of Science and Technology (HKUST) in 2007. In the last five years, he has published 74 SCI journal papers, >20 conference papers, three book chapters and two books. From 2012, his group has secured four funds from Australia Research Council.

Somaiyeh MahmoudZadeh completed her Master degree in Computer Science at National University of Malaysia. She is currently a Ph.D. student at the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. Her areas of research include computational intelligence, autonomy and decision making, situational awareness, and motion planning of autonomous underwater vehicles.