Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Steering control for underwater gliders*

You LIU[†1,2], Qing SHEN[2], Dong-li MA[1], Xiang-jiang YUAN[†‡2]

([1]*School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China*)

([2]*Institution of China Academy of Aerospace Aerodynamics, Beijing 100074, China*)

[†]E-mail: 542165262@qq.com; yuan_xj18@163.com

Received Nov. 23, 2016; Revision accepted Mar. 8, 2017; Crosschecked July 13, 2017

Abstract: Steering control for an autonomous underwater glider (AUG) is very challenging due to its changing dynamic characteristics such as payload and shape. A good choice to solve this problem is online system identification via in-field trials to capture current dynamic characteristics for control law reconfiguration. Hence, an online polynomial estimator is designed to update the yaw dynamic model of the AUG, and an adaptive model predictive control (MPC) controller is used to calculate the optimal control command based on updated estimated parameters. The MPC controller uses a quadratic program (QP) to compute the optimal control command based on a user-defined cost function. The cost function has two terms, focusing on output reference tracking and move suppression of input, respectively. Move-suppression performance can, at some level, represent energy-saving performance of the MPC controller. Users can balance these two competitive control performances by tuning weights. We have compared the control performance using the second-order polynomial model to that using the fifth-order polynomial model, and found that the former cannot capture the main characteristics of yaw dynamics and may result in vibration during the flight. Both processor-in-loop (PIL) simulations and in-lake tests are presented to validate our steering control performance.

Keywords: Autonomous underwater glider (AUG); Online system identification; Steering control; Adaptive control; Optimal control; Energy saving control; Processor-in-loop (PIL)

http://dx.doi.org/10.1631/FITEE.1601735 **CLC number:** TP242

1 Introduction

The autonomous underwater glider (AUG), as a kind of autonomous underwater vehicle (AUV), has become more and more popular, for example, the Seaglider, Slocum, and Spray (Eriksen *et al*., 2001; Rudnick *et al*., 2004), the Fòlaga (Alvarez *et al*., 2009), the Sterne (Phoemsapthawee *et al*., 2011), the Gliding Robotic Fish (Zhang *et al*., 2014; Zhang and Tan, 2015), and the Universiti Sains Malaysia (USM) hybrid-driven UG (Isa *et al*., 2014), for its advantages of long duration (ranging from weeks to months), long cover distance (more than 3000 km), and energy savings, when compared with propeller-powered AUVs in ocean research. However, its steering control is also more challenging than conventional AUVs. Due to relatively slow motion speed, the AUG is quite sensitive to environmental disturbance, such as surface waves and ocean currents. Thus, conventional control methods like proportional-integral-derivative (PID) linear quadratic regulator (LQR) controllers cannot provide both easy control implementation and high convergence speed for stabilization in the presence of continuously varying water disturbances. In addition, because the AUG dynamic is nonlinear, conventional linear-system-based PID controllers cannot dynamically compensate for unmodeled vehicle hydrodynamic forces. The unmodeled hydrodynamic forces may cause critical problems for steering control. For example, the glider Stern encountered 'counter-steering' behavior (Phoemsapthawee *et al*., 2013) and the first Stern model was lost

‡ Corresponding author

* Project supported by Beihang University and Institution of China Academy of Aerospace Aerodynamics

ORCID: Xiang-jiang YUAN, http://orcid.org/0000-0002-1862-1652

during sea trials due to unmodeled hydrodynamic forces. As the dynamic model underlies the design of its navigation, guidance, and control systems, any deviation from its nominal model would potentially degrade its performance or in the worst case, cause critical safety issues. Thus, a dynamic model based on measured data from experiments is more accurate and reliable than a purely theoretical model dependent on approximations and assumptions. Furthermore, the parameters in the AUG dynamic model vary when it has different payloads (net buoyancy) and glider geometry. For commercial real-world considerations, the AUG controller should be modular in design and support reconfiguration to fulfill the diverse requirements of end users (Eng *et al*., 2016). Therefore, the AUG control method should be adaptable to different payloads and shapes.

It is obvious that conventional PID and LQR controllers are inadequate to address the control difficulties mentioned earlier. One method to overcome such problems is online system identification, to update the yaw dynamic model for steering control. Online system identification has several advantages. The yaw dynamic model obtained through online system identification is more accurate and reliable than a purely theoretical yaw dynamic model, because its parameters are directly obtained from measured data. Online system identification works well whenever the payload (net buoyancy) and glider shape change. Compared with offline identification, online identification can reduce calculation complexity (Eng *et al*., 2016).

A newly manufactured AUG/AUV is usually programmed to perform a set of maneuvers under known excitation, and then the dynamic characteristics are obtained based on the behavior measured using on-board sensors. Caccia *et al*. (2000) and Mišković *et al*. (2011) used such an approach to capture desired dynamic features. Tiano *et al*. (2007) used an observer Kalman filter identification method to identify yaw dynamics of the Hammerhead AUV. Both simulation and experimental results were presented, but the online implementation of their algorithm was not discussed in detail. Similar work was done by Rentschler *et al*. (2006), where parameter estimation was performed offline using an optimization technique. Eng *et al*. (2016) proposed a state variable filter and recursive least square (SVF-RLS) estimator for online yaw dynamic identification, and compared it with the conventional offline identification method. The comparison showed that the SVF-RLS estimator is better in terms of prediction accuracy, computational cost, and training time. However, the gain-scheduled controller introduced in Eng *et al*. (2016) is conventional. Although this controller is adaptive to the linear-parameter-varying (LPV) model, its reference tracking performance is not optimized and the move suppression performance (energy saving) of the manipulated variable is ignored.

Our aim in this study is to propose an effective, optimal, energy-saving, and adaptive controller for underwater glider steering control. Compared with previous work mentioned earlier, critical features of our controller are described as follows:

1. Adaptivity. A polynomial estimator is designed to represent the current linear single-input, single-output (SISO) yaw dynamic model. During every control interval, the parameters are updated through online system identification and sent to the adaptive model predictive control (MPC) controller when they are available. Thus, the polynomial estimator is an LPV model that is adaptive to different payloads and glider geometries.

2. Reference tracking optimization. Reference tracking performance is the basic requirement for most controllers. A term focusing on this performance is involved in the cost function, and the adaptive MPC controller optimizes this cost function by using the KWIK (know what it knows) algorithm to determine the optimal manipulated variable (control input) sent to the servomotor during every control interval.

3. Energy conservation. Because steering control works during the whole voyage and the electric energy stored on an AUG is always limited, energy-saving performance is a critical aspect of the heading controller. We use an inner mass to adjust the lateral center of gravity (CG) location for heading control, and its severe vibration (see Fig. 17 in Section 5.2) may consume a large amount of additional electric power. A term representing input suppression is added to a cost function for energy-saving considerations. The controller minimizes the cost function to determine the optimal control command. Users can balance the reference tracking mentioned earlier and the move suppression here by tuning the weights

between them.

4. Universality. Although we discuss only steering control here, our method can be applied to other aspects, such as depth control. For simplicity, these other aspects are omitted in this paper.

5. Short design cycle. To find and solve potential problems in advance, we did processor-in-loop (PIL) simulations before the in-lake test. We compared the control performance using the second-order polynomial model (Eng *et al*., 2016) to that using the fifth-order polynomial model in PIL simulations, and found that the former cannot capture the main characteristics of yaw dynamics and may result in vibration during the simulation. Detecting and solving this problem before in-field experiments can shorten the whole AUG design cycle and avoid the waste of human, material, and financial resources on unnecessary experiments.

6. Effectivity. Both PIL simulation and in-lake tests have validated the effectivity of our method proposed in this paper.

2 Dynamics

2.1 Six-degree-of-freedom (DOF) motion modeling

Glider dynamic models have become more and more mature over the years. Leonard and Graver developed a dynamic model for a general underwater glider based on the mass point assumption and a simple parametric hydrodynamic model (Leonard and Graver, 2001; Graver, 2005). Wang *et al*. (2011) proposed a nonlinear dynamic model for a winged hybrid-driven underwater glider using linear and angular momentum equations based on multi-rigid body theory and a torpedo hydrodynamic model.

Three references are used in this study to clearly describe six-DOF motion dynamics. These frames are an inertia frame, a body frame, and a wind frame (Fig. 1).

Because the definitions of the three frames are commonly known in the AUG field, we do not introduce them in detail here. The rotation matrices to map vectors expressed in these frames are shown in matrix exponential form here:

$$\boldsymbol{R}_{\mathrm{IB}} = \mathrm{e}^{\hat{\boldsymbol{e}}_3\psi} \cdot \mathrm{e}^{\hat{\boldsymbol{e}}_2\theta} \cdot \mathrm{e}^{\hat{\boldsymbol{e}}_1\phi}, \tag{1}$$

$$\boldsymbol{R}_{\mathrm{BI}} = \boldsymbol{R}_{\mathrm{IB}}^{\mathrm{T}}, \tag{2}$$

$$\boldsymbol{R}_{\mathrm{BW}} = \mathrm{e}^{-\hat{\boldsymbol{e}}_2\alpha} \cdot \mathrm{e}^{\hat{\boldsymbol{e}}_3\beta}, \tag{3}$$

$$\boldsymbol{R}_{\mathrm{WB}} = \boldsymbol{R}_{\mathrm{BW}}^{\mathrm{T}}. \tag{4}$$

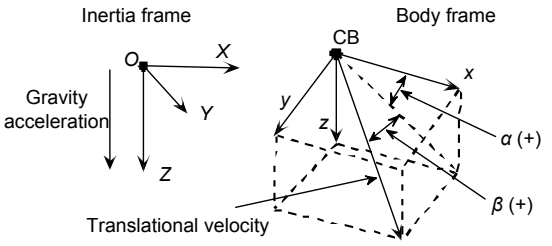

Fig. 1 Assignments of inertia, body, and wind frames

Positive directions of Euler angles ϕ, θ, and ψ satisfy the right-hand rule. Positive directions of α and β are shown in Fig. 1. Vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{e}_3$ have a standard Euclidean basis in the 3D space. Notation $\hat{\boldsymbol{e}}_i$ is the skew-symmetric cross-product matrix of $\boldsymbol{e}_i$. The angle of attack and angle of sideslip can be calculated in the following formula:

$$\beta = \arcsin(v / V),\ \alpha = \arctan(w / u), \tag{5}$$

where u, v, and w are components of the translational velocity of the glider in the body frame, and V is the magnitude of the translational velocity.

In our paper, the equations of motion, also known as the Euler–Newton equations (Phoemsapthawee *et al*., 2013), are developed for a glider whose CG can vary in the body reference frame. The Euler–Newton equations can be written in matrix form as

$$\begin{bmatrix} m\boldsymbol{I}_3 & -m\hat{\boldsymbol{R}}_{\mathrm{G}} \\ m\hat{\boldsymbol{R}}_{\mathrm{G}} & \boldsymbol{M}_{\mathrm{B}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{v}}_{\mathrm{B}} \\ \dot{\boldsymbol{\omega}}_{\mathrm{B}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{F}_{\mathrm{fic}} \\ \boldsymbol{M}_{\mathrm{fic}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\mathrm{ext}} \\ \boldsymbol{M}_{\mathrm{ext}} \end{bmatrix}, \tag{6}$$

where '^' is the skew-symmetric cross-product operator. $\boldsymbol{I}_3$ is the third-order identity matrix. $\boldsymbol{R}_{\mathrm{G}}$ represents the location of CG in the body frame, which can be written in component form as $[x_{\mathrm{G}}, y_{\mathrm{G}}, z_{\mathrm{G}}]^{\mathrm{T}}$. m is the total mass of the glider and is assumed to be constant. In this study, x_{G} and z_{G} are constant and y_{G} is regarded as a control input that is used to control the yaw angle. Its value is determined by the controller, which will be introduced in Section 4. v_{B} represents translational

velocity, which can be written as $[u, v, w]^{\mathrm{T}}$ in the body frame. ω_{B} represents the angular velocity, which can be written as $[p, q, r]^{\mathrm{T}}$ in the body frame. $\boldsymbol{M}_{\mathrm{B}}$ is the matrix of inertia determined by the CG location. $\boldsymbol{F}_{\mathrm{fic}}$ and $\boldsymbol{M}_{\mathrm{fic}}$ are fictitious force and moment, respectively:

$$\begin{bmatrix} \boldsymbol{F}_{\mathrm{fic}} \\ \boldsymbol{M}_{\mathrm{fic}} \end{bmatrix} = \begin{bmatrix} m\hat{\boldsymbol{\omega}}_{\mathrm{B}}\boldsymbol{v}_{\mathrm{B}} + m\hat{\boldsymbol{\omega}}_{\mathrm{B}}(\hat{\boldsymbol{\omega}}_{\mathrm{B}}\boldsymbol{R}_{\mathrm{G}}) \\ \hat{\boldsymbol{\omega}}_{\mathrm{B}}(\boldsymbol{M}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}}) + m\hat{\boldsymbol{R}}_{\mathrm{G}}(\hat{\boldsymbol{\omega}}_{\mathrm{B}}\boldsymbol{v}_{\mathrm{B}}) \end{bmatrix}. \tag{7}$$

The external force and moment applied on the glider consist of hydrodynamic force and moment, gravity, and buoyancy:

$$\begin{bmatrix} \boldsymbol{F}_{\mathrm{ext}} \\ \boldsymbol{M}_{\mathrm{ext}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\mathrm{h}} + \boldsymbol{B} \\ \boldsymbol{M}_{\mathrm{h}} + m\hat{\boldsymbol{R}}_{\mathrm{G}}\boldsymbol{g} \end{bmatrix}. \tag{8}$$

$\boldsymbol{F}_{\mathrm{h}}$ and $\boldsymbol{M}_{\mathrm{h}}$ are hydrodynamic force and moment, respectively, $\boldsymbol{B}$ represents net buoyancy, and $\boldsymbol{g}$ is the acceleration of gravity. Using the rotation matrices (Eqs. (1)–(4)) mentioned earlier, $\boldsymbol{B}$ and $\boldsymbol{g}$ are equivalent to $\boldsymbol{R}_{\mathrm{BI}}\boldsymbol{B}_{\mathrm{ine}}$ and $\boldsymbol{R}_{\mathrm{BI}}\boldsymbol{g}_{\mathrm{ine}}$, respectively. $\boldsymbol{B}_{\mathrm{ine}}$ and $\boldsymbol{g}_{\mathrm{ine}}$ can be written as $[0, 0, B]^{\mathrm{T}}$ and $[0, 0, 9.81]^{\mathrm{T}}$ in the inertia frame, respectively. B is regarded as a control variable given before the motion simulation. Hydrodynamic force and moment are presented as follows:

$$\begin{bmatrix} \boldsymbol{F}_{\mathrm{h}} \\ \boldsymbol{M}_{\mathrm{h}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\mathrm{ine}} \\ \boldsymbol{M}_{\mathrm{ine}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{F}_{\mathrm{vis}} \\ \boldsymbol{M}_{\mathrm{vis}} \end{bmatrix}. \tag{9}$$

$\boldsymbol{F}_{\mathrm{ine}}$ and $\boldsymbol{M}_{\mathrm{ine}}$ are inertial hydrodynamic force and moment around CB, respectively. $\boldsymbol{F}_{\mathrm{vis}}$ and $\boldsymbol{M}_{\mathrm{vis}}$ are viscid hydrodynamic force and moment around CB, respectively. Hydrodynamic force and moment calculations will be introduced in detail in the next section. Note that all the vectors above are in the body frame. For our glider, the total mass is constant and the matrix of inertia is determined by the gravity location. Because the out-body geometry does not vary, the added inertial matrix and center of buoyancy do not change with time.

To describe the glider motion in the inertia frame, we need two other equations:

$$\dot{\boldsymbol{b}} = \boldsymbol{R}_{\mathrm{IB}}\boldsymbol{v}_{\mathrm{B}}, \tag{10}$$

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{R}_{\boldsymbol{\Omega}\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}}. \tag{11}$$

$\boldsymbol{b}$ is the displacement vector from O to CB, which can be written in component form as $[x, y, z]^{\mathrm{T}}$ in the inertia frame. $\boldsymbol{\Omega}$ is the Euler angle vector written in component form as $[\varphi, \theta, \psi]^{\mathrm{T}}$. $\boldsymbol{R}_{\boldsymbol{\Omega}\mathrm{B}}$ is the rotation matrix mapping angular velocity expressed in the body frame as an Euler angular velocity vector. $\boldsymbol{R}_{\boldsymbol{\Omega}\mathrm{B}}$ can be expressed in matrix form as follows:

$$\boldsymbol{R}_{\boldsymbol{\Omega}\mathrm{B}} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}.$$

Eqs. (6), (10), and (11) constitute our six-DOF motion dynamic model.

The dynamic model we have derived is nonlinear and can be classified as a differential algebraic equation (DAE) system. The DAE system consists of 12 component differential equations with 12 time-dependent variables. The variables can be written in vector form as $\boldsymbol{x}(t)=[x, y, z, u, v, w, \phi, \theta, \psi, p, q, r]^{\mathrm{T}}$. Then the dynamic model can be simplified to

$$\boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), t) = 0. \tag{12}$$

2.2 Hydrodynamic model

2.2.1 Viscid hydrodynamic model

In this study, we use a classic submarine hydrodynamic model introduced in our previous paper (Liu *et al*., 2016) to calculate viscid force and moment acting on an underwater glider. The model is based on Taylor series expansion of hydrodynamic forces and moments in suitable initial conditions. If the angle of attack, angle of sideslip, speed, and angular velocity are small, higher-order terms in the Taylor series can be ignored and the model can be simplified to

$$\begin{aligned} \boldsymbol{F}_{\mathrm{h}} &= \begin{bmatrix} C_X^{\alpha} \\ C_Y^{\beta} + C_Y^{P} + C_Y^{R} \\ C_Z^{\alpha} + C_Z^{Q} \end{bmatrix} \cdot 0.5\rho V^2 S \\ &= \begin{bmatrix} X_{\alpha}^{0} + X_{\alpha}^{2}\alpha^2 \\ Y_{\beta}^{1}\beta + Y_{P}^{1}P + Y_{R}^{1}R \\ Z_{\alpha}^{1}\alpha + Z_{\alpha}^{0} + Z_{Q}^{1}Q \end{bmatrix} \cdot 0.5\rho V^2 S, \end{aligned} \tag{13}$$

$$\boldsymbol{M}_{\mathrm{h}}=\begin{bmatrix} C_K^P \\ C_M^\alpha+C_M^Q \\ C_N^\beta+C_N^P+C_N^R \end{bmatrix}\cdot 0.5\rho V^2 SL =\begin{bmatrix} K_P^1 P \\ M_\alpha^1\alpha+Y_Q^1 Q \\ N_\beta^1\beta+N_P^1 P+N_R^1 R \end{bmatrix}\cdot 0.5\rho V^2 SL. \tag{14}$$

Parameters ρ, V, S, and L represent the density of water, velocity, wing area, and length of the glider, respectively. $\boldsymbol{F}_{\mathrm{h}}$ and $\boldsymbol{M}_{\mathrm{h}}$ represent hydrodynamic force and moment about CB, respectively. Here, all the hydrodynamic forces and moments are described in the body frame. The angle of attack is α. The side-slip angel is β. The definitions of P, Q, and R are given as follows:

$$P=pL/V, \tag{15}$$

$$Q=qL/V, \tag{16}$$

$$R=rL/V. \tag{17}$$

Parameters p, q, and r represent the angular velocities around the x, y, and z axes, respectively. The remaining terms in Eqs. (13) and (14) are dimensionless coefficients, which can be estimated using the computational fluid dynamics (CFD) method introduced by Liu *et al*. (2016). The values of all hydrodynamic coefficients are listed in Table 1. The profile of our glider and main dimensions are presented in Fig. 2.

Table 1 Viscid hydrodynamic coefficients of the underwater glider

Hydrodynamic parameter	Value	Hydrodynamic parameter	Value
X_α^0	−0.05869	Z_Q^1	−0.7545
X_α^2	3.971	K_P^1	−0.1359
Y_β^1	−0.8390	M_α^1	−0.4273
Y_P^1	−0.01076	M_Q^1	−0.1168
Y_R^1	0.3052	N_P^1	0.006628
Z_α^0	0.001184	N_R^1	−0.1658
Z_α^1	−5.630	N_β^1	0.02747

The main physical parameters are listed in Table 2. Because our glider has approximately two symmetries, the non-diagonal elements in the inertia matrix $\boldsymbol{M}_{\mathrm{B}}$ are approximately equivalent to zero. Diagonal elements I_{xx}, I_{yy}, and I_{zz} represent moments of inertia around the x, y, and z axes, respectively. They are determined by the CG location. For our glider, the CG location has quite small changes in distance; for example, the absolute values of x_{G}, y_{G}, and z_{G} are less than 0.005 m in this study. We can ignore the influence of the CG location and use CATIA software to estimate them at $\boldsymbol{R}_{\mathrm{G}}=[0, 0, 0]^{\mathrm{T}}$. During motion simulation, the inertia matrix does not vary.

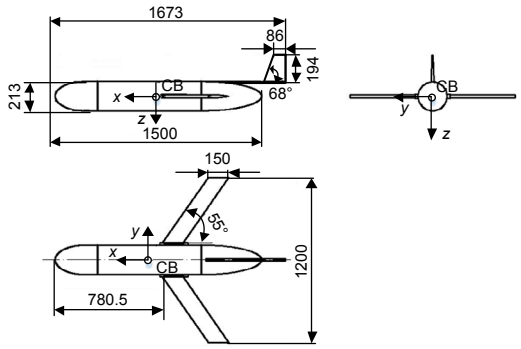

Fig. 2 Three views of the underwater glider (unit of dimensions is millimeter, and unit of angles is degree)

Table 2 Physical parameters of the underwater glider

Hydrodynamic parameter	Value
L	1.6730 m
m	49.1930 kg
S	0.15 m^2
ρ	998.2 $\mathrm{kg/m}^3$
I_{xx}	0.8265 $\mathrm{kg\cdot m}^2$
I_{yy}	7.5484 $\mathrm{kg\cdot m}^2$
I_{zz}	8.0888 $\mathrm{kg\cdot m}^2$

2.2.2 Inertia hydrodynamic model

Based on potential flow theory, the inertia force and moment around CB can be described as

$$\begin{bmatrix} \boldsymbol{F}_{\mathrm{ine}} \\ \boldsymbol{M}_{\mathrm{ine}} \end{bmatrix}=-\boldsymbol{M}_{\mathrm{add}}\begin{bmatrix} \dot{\boldsymbol{v}}_{\mathrm{B}} \\ \dot{\boldsymbol{\omega}}_{\mathrm{B}} \end{bmatrix}. \tag{18}$$

$\boldsymbol{M}_{\mathrm{add}}$ is the added inertia matrix, which involves 36 elements. Based on potential flow theory, the matrix is symmetrical and there are only 21 independent elements. If the glider has two symmetrical planes, only eight independent elements are left non-zero. Our glider has left/right symmetry and approximate

up/down symmetry. Then the remaining independent non-zero elements in the added inertia matrix are exhibited as follows:

$$\lambda_{11}, \lambda_{22}, \lambda_{33}, \lambda_{44}, \lambda_{55}, \lambda_{66}, \lambda_{26}, \lambda_{35}.$$

The methods to estimate these elements consist of geometry-based estimation (Chen, 1981; Shi, 1995; Li, 1999) and a potential flow theory based numerical approach (Li *et al*., 2010). Actually, geometry-based estimation is not always accurate in comparison with experimental data, and this method is limited to some special simple shapes. Xiao (2014) used CFD to calculate inertial hydrodynamic coefficients and compared the results with experimental data. They concluded that the CFD result error is within 6%. In light of the knowledge mentioned earlier, we choose the CFD method to calculate elements in the added mass matrix.

To calculate λ_{11}, λ_{22}, λ_{33}, λ_{26}, and λ_{35}, the boundary condition of the glider surface is set as a no-slip wall and the out boundary of the calculation domain is set as a velocity-inlet with velocity $\boldsymbol{V}$=[−0.0001, 0, 0]$^{\mathrm{T}}$ expressed in the inertia frame (Fig. 3). The viscous model is inviscid based on potential flow theory, so the velocity that is set on the velocity-inlet boundary actually does not influence the calculation results. The velocity set on the velocity-inlet boundary is for convergence considerations. We use the user-defined function to define the mesh movement of the calculation domain, and the translational velocity is expressed as follows:

$$v_x = at,\ v_y = at,\ v_z = at, \tag{19}$$

where v_x, v_y, and v_z are components of velocity in the inertia frame and a represents the component of velocity acceleration in the inertia frame. In Fluent, the mesh movement of the calculation domain does not influence the velocity-inlet boundary condition, but the wall moving together with the whole domain has the same velocity as the calculation domain. We use a pressure-based transient solver to simulate the unsteady flow, and other unmentioned options remain at default in Fluent. We simulate the potential flow three times at a=0.05, 0.1, and 0.15 m/s^2 with a time step of 0.0001 s and obtain elements λ_{11}, λ_{22}, λ_{33}, λ_{26}, and λ_{35} (Table 3). Convergence is met when the relative error between two time steps is less than 10^{-4}. Note that the values in Table 3 are averages of the three simulations.

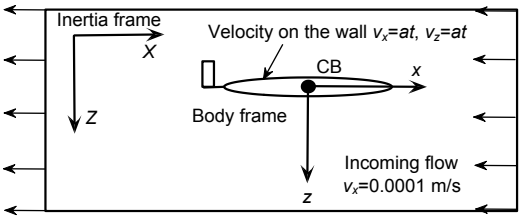

Fig. 3 Illustration of boundary conditions in the *XY* plane to calculate inertial coefficients

To calculate λ_{33}, λ_{44}, and λ_{66}, the approach is almost the same as stated earlier. However, the domain zone is rotational and the angular velocity can be described as follows:

$$\omega_x = \varepsilon t,\ \omega_y = \varepsilon t,\ \omega_z = \varepsilon t, \tag{20}$$

where ω_x, ω_y, and ω_z are components of angular velocity in the inertia frame and ε represents the component of angular velocity acceleration in the inertia frame. We simulate the potential flow three times at ε=0.05, 0.1, and 0.15 rad/s^2 with a time step of 0.0001 s and obtained elements λ_{33}, λ_{44}, and λ_{66} (Table 3). Note that the values in Table 3 are averages of the three simulations.

Table 3 Elements in the added mass matrix

Inertial hydrodynamic coefficient	Value
λ_{11}	2.13 kg
λ_{22}	44.77 kg
λ_{26}	−1.73 kg·m
λ_{33}	63.13 kg
λ_{35}	4.75 kg·m
λ_{44}	1.54 kg·m^2
λ_{55}	6.38 kg·m^2
λ_{66}	6.46 kg·m^2

2.3 Yaw dynamics

We use MATLAB® to linearize the six-DOF motion dynamic model in Eq. (12) at an equilibrium point, where $\boldsymbol{R}_{\mathrm{G}}$=[0.0023, 0, 0.005]$^{\mathrm{T}}$ m, B=4.9 N, and [x, y, z, u, v, w, φ, θ, ψ, p, q, r]$^{\mathrm{T}}$=[321.58 m, 0 m, 255.86 m, 0.83 m/s, 0 m/s, 0.011 m/s, 0 rad, −0.65 rad,

0 rad, 0 rad/s, 0 rad/s, 0 rad/s]$^{\mathrm{T}}$. The equilibrium point is not unique. Any equilibrium point near the point used in this study is reasonable. We are most concerned with the yaw dynamic structure instead of the parameters in the yaw dynamic model. Treating y_{G} as input and ψ as output, the linearized model can be converted to the transfer function as follows:

$$Y(s)=F(s)U(s), \tag{21}$$

where $F(s)=(-25.38s^3-253.2s^2-140.6s-2.998)/(s^5+12.55s^4+27.34s^3+13.69s^2+1.194s)$. We discretize the continuous time dynamic linear system using zeroth-order hold on the inputs with sample time T_{s}=0.1 s. The result is as follows:

$$Y(z)=F(z)U(z), \tag{22}$$

where $F(z)=(-0.117z^4+0.1655z^3+0.05169z^2-0.1399z+0.03964)/(z^5-4.128z^4+6.677z^3-5.255z^2+1.991z-0.285)$. This is a fifth-order polynomial model. It can be expressed in the following difference equation form:

$$y(k)=\sum_{i=1}^{5}b_iu(k-i)-\sum_{i=1}^{5}a_iy(k-i), \tag{23}$$

where a_1=−4.128, a_2=6.677, a_3=−5.255, a_4=1.991, a_5=−0.285, b_1=−0.117, b_2=0.1655, b_3=0.05169, b_4=−0.13991, and b_5=0.03964. $y(k)$ and $u(k)$ represent the output and input of the dynamic system at the kth sample time, respectively. As we know, a_i and b_i vary when we linearize the dynamic model at different equilibrium points, but the structure of the difference equation changes slightly. We use this structure to represent yaw dynamics and estimate a_i and b_i using an online estimator, which will be introduced in Section 3. These parameters can be expressed in vector form as $\hat{\boldsymbol{\theta}}(k)=[-a_{1k}, -a_{2k}, -a_{3k}, -a_{4k}, -a_{5k}, b_{1k}, b_{2k}, b_{3k}, b_{4k}, b_{5k}]^{\mathrm{T}}$ at the kth sample time. Let $\boldsymbol{A}_k=[a_{1k}, a_{2k}, a_{3k}, a_{4k}, a_{5k}]^{\mathrm{T}}$ and $\boldsymbol{B}_k=[b_{1k}, b_{2k}, b_{3k}, b_{4k}, b_{5k}]$. Thus, $\hat{\boldsymbol{\theta}}(k)=[-A^{\mathrm{T}}, B^{\mathrm{T}}]$. Another simpler structure is used in this study to compare its performance to that of the structure in Eq. (23). The simpler structure is a second-order polynomial model proposed in Eng *et al*. (2016), and it can be expressed in difference form:

$$y(k)=\sum_{i=1}^{2}b_iu(k-i)-\sum_{i=1}^{2}a_iy(k-i). \tag{24}$$

In the next section, we will introduce how the online estimator works using the polynomial model derived earlier.

3 Online estimator

The following set of equations summarizes the online estimator algorithm (Eng *et al*., 2016) used in this study:

$$\hat{\boldsymbol{\theta}}(k)=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{K}(k)[\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k)], \tag{25}$$

$$\hat{\boldsymbol{y}}(k)=\boldsymbol{\Psi}^{\mathrm{T}}(k)\hat{\boldsymbol{\theta}}(k-1), \tag{26}$$

$$\boldsymbol{K}(k)=\frac{\boldsymbol{P}(k-1)\boldsymbol{\Psi}(k)}{1+\boldsymbol{\Psi}^{\mathrm{T}}(k)\boldsymbol{P}(k-1)\boldsymbol{\Psi}(k)}, \tag{27}$$

$$\boldsymbol{P}(k)=[1-\boldsymbol{K}^{T}(k)\boldsymbol{\Psi}(k)]\boldsymbol{P}(k-1). \tag{28}$$

$\hat{\boldsymbol{\theta}}(k)$ represents the parameters vector estimated at the kth sample time (sample time is 0.1 s in this study). $\boldsymbol{\Psi}(k)$ is the regression vector at the kth sample time determined by previous input and output signals and can be expressed in vector form as [$y(k-1)$, $y(k-2)$, $y(k-3)$, $y(k-4)$, $y(k-5)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, $u(k-5)$]$^{\mathrm{T}}$. $\boldsymbol{P}(k)$ is the covariance matrix at the kth sample time.

This algorithm is entirely specified by the sequence of data $y(k)$, the gradient (regression vector) $\boldsymbol{\Psi}(k)$, the initial conditions $\boldsymbol{\theta}(t=0)$ (initial guess of the parameters), and $\boldsymbol{P}(t=0)$ (covariance matrix that indicates parameter errors). In this study, $y(k)$ is recorded by a digital compass every 0.1 s in experiments (in Section 6) or read from the host computer in PIL simulations (in Section 5), and it is noise-free by using a built-in low-pass finite impulse response (FIR) filter in the experiment. In the experiment, $u(k)$ is the control command sent to the servomotor to control the CG location y_{G} every 0.1 s. In PIL simulation, $u(k)$ is the control command sent to the host computer to solve the six-DOF motion dynamic model. It is assumed that the CG location y_{G} is the same as the commanded location sent to the servomotor in the experiment. This assumption holds because the response time of the servomotor is many times faster

than the yaw dynamics (Eng *et al*., 2016). $\boldsymbol{\Psi}(k)$ is determined by previous values of $u(k)$ and $y(k)$. An initial guess of $\boldsymbol{\theta}(t=0)$ is vector $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^{\mathrm{T}}$. $\boldsymbol{P}(t=0)$ is equal to $1000\boldsymbol{I}_{10}$, where $\boldsymbol{I}_{10}$ is the 10th-order identity matrix.

In Section 4, we introduce how to compute the control command $u(k)$ based on the estimated parameters obtained by the online estimator during every control interval.

4 Model predictive control controller

The MPC controller uses the parameters computed by the online estimator at the kth sample time to predict the outputs for the next p time steps. p is called the prediction horizon, always set as five in this study. The method is shown here:

$$y(k+i)=\sum_{j=1}^{5}b_{jk}u(k+i-j)-\sum_{j=1}^{5}a_{jk}y(k+i-j), \quad (29)$$

where i varies from 1 to p. a_{jk} and b_{jk} represent the parameters estimated at the kth sample time. $y(k+i)$ is the predicted value of the output signal. Because the inputs and outputs before the kth sample time are already known, it is worth noting that the predicted output is a function of the predicted inputs $\boldsymbol{z}_k=[u(k), u(k+1), \ldots, u(k+p-1)]^{\mathrm{T}}$ only. To compute the optimal values of the predicted inputs, we minimize a cost function that represents the performance of fast reference tracking and input suppression to determine the plant input signal. This cost function consists of two terms. The first term, called the output reference tracking function, is shown here:

$$J_y(z_k)=\sum_{i=1}^{p}\left[\frac{w_1(r(k+i)-y(k+i))}{s_1}\right]^2, \quad (30)$$

where k is the current control interval (kth sample time), p is the prediction horizon (number of intervals), $\boldsymbol{z}_k$ is the input signal, given by $\boldsymbol{z}_k=[u(k), u(k+1), \ldots, u(k+p-1)]^{\mathrm{T}}$, $y(k+i)$ is the predicted value of output at the ith prediction horizon step, $r(k+i)$ is the reference value at the ith prediction horizon step, which is given by users, w_1 is the tuning weight for plant output at the ith prediction horizon step (dimensionless), and s_1 is the scale factor for plant output in engineering units. In this study, s_1 is always set as 2π rad.

The values p and w_1 are controller specifications and are constant. The controller receives $r(k+i)$ values for the entire prediction horizon. The controller uses Eq. (29) to predict the plant outputs. Thus, J_y is a function of $\boldsymbol{z}_k$ only.

As for our underwater glider, move suppression performance is also a critical factor for energy savings. Because severe vibration of the inner movable mass may lead to additional energy consumption and stability problems, the move-suppression function is used to restrain the variance of the control command sent to the servomotor. We use a cost function to represent this performance:

$$J_{\Delta u}(z_k)=\sum_{i=1}^{p-1}\left[\frac{w_2(u(k+i)-u(k+i-1))}{s_2}\right]^2, \quad (31)$$

where k is the current control interval (kth sample time), p is the prediction horizon (number of intervals), $\boldsymbol{z}_k$ is the input signal, given by $\boldsymbol{z}_k=[u(k), u(k+1), \ldots, u(k+p-1)]^{\mathrm{T}}$, $u(k+i)$ is the predicted value of output at the ith prediction horizon step, and $r(k+i)$ is the reference value at the ith prediction horizon step, which is given by users, w_2 is the tuning weight for plant output at the ith prediction horizon step (dimensionless), and s_2 is the scale factor for plant input in engineering units. In this study, s_2 is always set as 0.01 m.

The values p and w_2 are controller specifications and are constant. Thus, $J_{\Delta u}$ is a function of $\boldsymbol{z}_k$ only.

In summary, the cost function we finally use is the sum of the two terms mentioned earlier, each focusing on a particular aspect of controller performance, as follows:

$$J(z_k)=J_y(z_k)+J_{\Delta u}(z_k). \quad (32)$$

This function is determined by $\boldsymbol{z}_k$ only. $\boldsymbol{z}_k$ is the input decision; it will be computed using a quadratic programming (QP) minimization method. As described previously, each term includes weights that help balance competing objectives, such as reference

tracking and move suppression.

For our glider, the CG location y_G (control input) is limited to a certain scope for a limited glider inner space, and the input rate is also limited for the limited response rate of the servomotor. Thus, we have the following constraint equations at every control interval:

$$-0.005\ \text{m} \le u(k+i) \le 0.005\ \text{m}, \tag{33}$$

$$-0.0001\ \text{m} \le u(k+i) - u(k+i-1) \le 0.0001\ \text{m}, \tag{34}$$

where i ranges from 0 to $p-1$.

The problem of minimizing cost function (32) under the constraint equations can be converted to the standard QP form. Now we introduce the process in detail. Using Eq. (29), the predicted output values can be written as

$$\begin{aligned}\begin{bmatrix} r(k+i)-y(k+i) \\ \vdots \\ r(k+p)-y(k+p) \end{bmatrix} &= \boldsymbol{E}\begin{bmatrix} u(k) \\ \vdots \\ u(k+p-1) \end{bmatrix} + \boldsymbol{E}_0 \\ &= \boldsymbol{E}\boldsymbol{z}_k + \boldsymbol{E}_0,\end{aligned} \tag{35}$$

where $\boldsymbol{E}$ is a $p\times p$ matrix determined by known values $y(k)$, $y(k-1)$, $y(k-2)$, $y(k-3)$, $y(k-4)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, and $\hat{\boldsymbol{\theta}}(k)$ at the kth sample time. $\boldsymbol{E}_0$ is a $p\times 1$ matrix determined by known values $y(k)$, $y(k-1)$, $y(k-2)$, $y(k-3)$, $y(k-4)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, and $\hat{\boldsymbol{\theta}}(k)$ at the kth sample time.

Now, we introduce the method for calculating the elements in matrices $\boldsymbol{E}$ and $\boldsymbol{E}_0$. Using Eq. (29), we have $y(k+1)=-a_{1k}u(k)-a_{2k}u(k-1)-a_{3k}u(k-2)-a_{4k}u(k-3)-a_{5k}u(k-4)+b_{1k}y(k)+b_{2k}y(k-1)+b_{3k}y(k-2)+b_{4k}y(k-3)+b_{5k}y(k-4)$. Then, the elements in the first row of matrix $\boldsymbol{E}$ can be expressed as $E_{11}=a_{1k}$, $E_{12}=0$, $E_{13}=0$, $E_{14}=0$, and $E_{15}=0$. The element in the first row of $\boldsymbol{E}_0$ can be expressed as $E_{011}=a_{2k}u(k-1)+a_{3k}u(k-2)+a_{4k}u(k-3)+a_{5k}u(k-4)-b_{1k}y(k)-b_{2k}y(k-1)-b_{3k}y(k-2)-b_{4k}y(k-3)-b_{5k}y(k-4)+r(k+1)$. The rest of the elements in $\boldsymbol{E}$ and $\boldsymbol{E}_0$ can be obtained in the same way as shown earlier.

To express Eq. (31) in matrix form, we have the following formula:

$$\begin{aligned}\begin{bmatrix} u(k)-u(k-1) \\ \vdots \\ u(k+p-1)-u(k+p-2) \end{bmatrix} &= \boldsymbol{F}\begin{bmatrix} u(k) \\ \vdots \\ u(k+p-1) \end{bmatrix} + \boldsymbol{F}_0 \\ &= \boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_0.\end{aligned} \tag{36}$$

$\boldsymbol{F}$ can be written as follows:

$$\boldsymbol{F} = \boldsymbol{I}_{p\times p} - \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}_{p\times p}, \tag{37}$$

$$\boldsymbol{F}_0 = -\begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{p\times 1}. \tag{38}$$

$\boldsymbol{I}_{p\times p}$ is a $p\times p$ identity matrix.

Substituting Eqs. (35) and (36) into Eq. (32), the cost function can be written as

$$\begin{aligned}J(\boldsymbol{z}_k) = &(\boldsymbol{E}\boldsymbol{z}_k + \boldsymbol{E}_0)^{\mathrm{T}}\boldsymbol{w}_y^2(\boldsymbol{E}\boldsymbol{z}_k + \boldsymbol{E}_0) \\ &+ (\boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_0)^{\mathrm{T}}\boldsymbol{w}_{\Delta u}^2(\boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_0),\end{aligned} \tag{39}$$

where $\boldsymbol{w}_y$ is equal to $w_1/s_1\boldsymbol{I}_{p\times p}$ and $\boldsymbol{w}_{\Delta u}$ is equal to $w_2/s_2\boldsymbol{I}_{p\times p}$.

Furthermore, Eq. (36) can be simplified to

$$J(\boldsymbol{z}_k) = \boldsymbol{z}_k^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z}_k + 2\boldsymbol{f}^{\mathrm{T}}\boldsymbol{z}_k + \text{constant}. \tag{40}$$

Thus, we can minimize the expression as follows to minimize the cost function above:

$$\min_{\boldsymbol{z}_k}\ \boldsymbol{f}^{\mathrm{T}}\boldsymbol{z}_k + \frac{1}{2}\boldsymbol{z}_k^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z}_k \tag{41}$$

such that

$$\boldsymbol{A}\boldsymbol{z}_k \le \boldsymbol{b}, \tag{42}$$

where $\boldsymbol{z}_k=[u(k),\ u(k+1),\ \ldots,\ u(k+p-1)]^{\mathrm{T}}$ is the input decision, and $\boldsymbol{H}$ is the Hessian matrix. $\boldsymbol{A}$ and $\boldsymbol{b}$ are determined by linear constraint coefficients in Eqs. (33) and (34). $\boldsymbol{f}^{\mathrm{T}}$, $\boldsymbol{H}$, $\boldsymbol{A}$, and $\boldsymbol{b}$ are all known and the

controller computes these matrices at the beginning of each control instant.

If ignoring constraint (42), the minimization problem has the following analytical solution:

$$\boldsymbol{z}_* = -\boldsymbol{H}^{-1}\boldsymbol{f}. \tag{43}$$

If constraint (42) is considered, we use the classical KWIK algorithm (Schmid and Biegler, 1994) to solve the QP problem described in Eqs. (41) and (42). In the very first control step, KWIK uses a cold start, in which the initial guess is the unconstrained solution described in Eq. (43). If this solution satisfies the constraints, it is the optimal QP solution, $\boldsymbol{z}_*$, and the algorithm terminates. Otherwise, at least one of the linear inequality constraints must be satisfied as an equality. In this case, KWIK uses an efficient, numerically robust strategy to determine the active constraint set satisfying the standard optimality conditions. In the following control steps, KWIK uses a warm start. In this case, the active constraint set determined at the previous control step becomes the initial guess for the next.

Note that the actual control command sent to the servomotor at the kth control interval is $u(k)$ only and that the remaining values in vector $\boldsymbol{z}_k=[u(k), u(k+1), \ldots, u(k+p-1)]^{\mathrm{T}}$ are all ignored.

Although KWIK is robust, we have considered the following special situations:

The search for the active constraint set is an iterative process. If the iterations reach a problem-dependent maximum, the algorithm terminates.

Because the QP problem includes hard constraints (Eqs. (32) and (33)), these constraints might be infeasible (impossible to satisfy). If the algorithm detects infeasibility, it terminates immediately.

In the last two situations, with an abnormal outcome to the search, the controller will retain the last successful control command.

5 Processor-in-loop validation

In Sections 2–4, we have introduced our six-DOF dynamic model, online estimator, and adaptive MPC controller in detail. From now on, we use the PIL technology to validate their performance. PIL means that the online estimator and MPC controller operate in the microcomputer, while the six-DOF dynamic model works in the host computer. The microcomputer is a TI DSP-F28335 chip based board which can send and receive data from the host computer using an RS232 cable (Fig. 4).

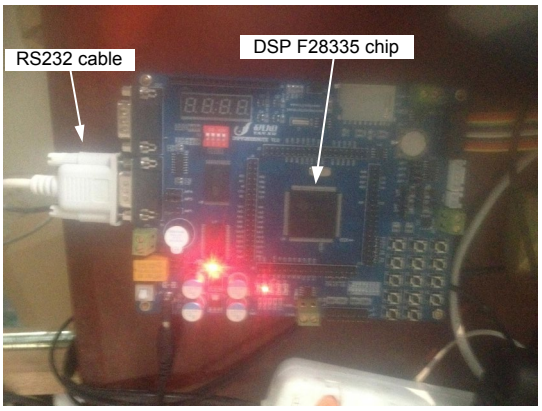

Fig. 4 Exhibition of the F28335-based board

On the host side (PC), MATLAB simulates the six-DOF dynamic model and sends the current yaw angle to the microcomputer through the RS232 cable. On the target board side, the MATLAB Embedded Coder Toolbox can download the algorithms of the estimator and MPC controller into the chip through Code Composer Studio 5.5 (CCS 5.5) software and they operate on the chip independently. When the current yaw angle is available from the host side, the chip reads the data, computes the current plant input, and sends it back to the host side. The data exchange model between the host and target side is the serial communication interface (SCI) and the baud rate is set as 11520. Our PIL simulations consist of three steps:

1. Testing the performance of the online estimator. We monitor the time history of the estimated parameters using the given excitation input and test the convergence performance. At the same time, we compare the output prediction based on the estimated parameters and Eq. (23) with the six-DOF dynamic model output to validate the correctness of our online estimator. The estimated parameters are the initial condition for the next step. During the simulation in this step, the MPC controller is shut down.

2. Testing the MPC controller performance including reference tracking and move suppression based on the parameters obtained in the previous step. We introduce how to balance these two types of

competing performance by tuning weights, as mentioned in Section 4.

3. Testing if the second-order polynomial structure (Eng *et al*., 2016) used by previous researchers is powerful enough to capture the main dynamic features and can be applied to our underwater glider.

5.1 Online estimator validation

In this section, we validate the performance of the online estimator. Fig. 5 shows the data flow on the host side and target side.

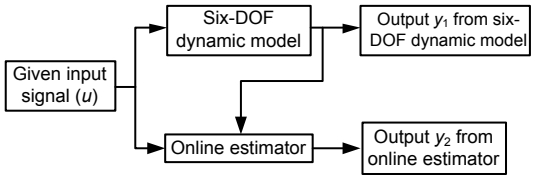

Fig. 5 Data flow of the processor-in-loop simulation

The online estimator receives and sends data every 0.1 s. The six-DOF dynamic model simulates glider movement using the fourth-order Runge–Kutta method with the initial condition $\boldsymbol{x}_0(t)$=[0, 0, 0, 0.1 m/s, 0, 0, 0, 0, 0, 0, 0, 0]$^{\mathrm{T}}$. x_{G}=0.0013 m, z_{G}=0.005 m, and B=4.9 N. The time history of the given input signal is shown in Fig. 6.

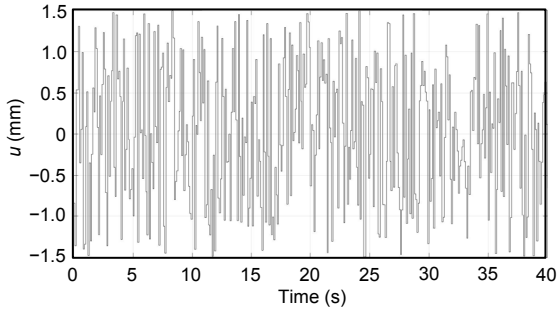

Fig. 6 Given excitation of the input signal

The given excitations are uniformly distributed random numbers ranging from −1.5 mm to 1.5 mm. Under such excitation, the parameters computed by the online estimator are shown in Figs. 7 and 8.

As shown in Figs. 7 and 8, all the parameters converge quickly from the initial guess $\boldsymbol{\theta}(t=0)$=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$^{\mathrm{T}}$, which indicates that our online estimator has good convergence performance. The actual values of all the parameters at the end of the simulation are listed in Table 4. In the next section, the controller and online estimator use these values as the initial condition.

As shown in Fig. 5, the output can be computed using the six-DOF motion dynamic model in MATLAB or using the linear polynomial dynamic model expressed in Eq. (23). The results of these two outputs and the error between them are shown in Fig. 9.

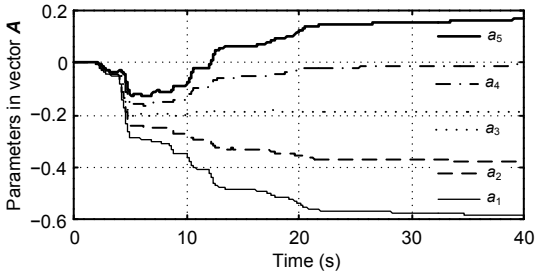

Fig. 7 Time history of parameters in vector *A*

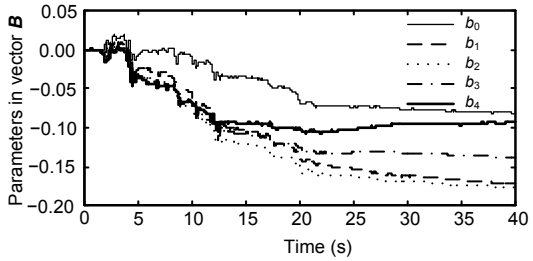

Fig. 8 Time history of parameters in vector *B*

Table 4 Estimated parameters in vectors *A* and *B* through processor-in-loop simulation

Parameter	Value	Parameter	Value
a_1	−0.58	b_0	−0.080
a_2	−0.37	b_1	−0.17
a_3	−0.18	b_2	−0.17
a_4	−0.0095	b_3	−0.13
a_5	0.16	b_4	−0.093

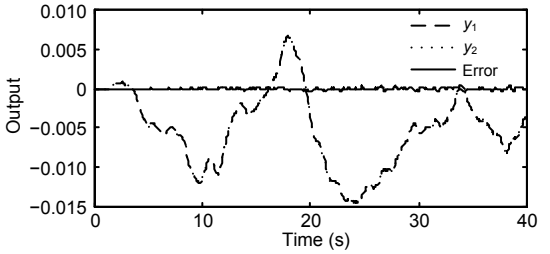

Fig. 9 Comparison between outputs y_1 and y_2 during the processor-in-loop simulation

As shown in Fig. 9, the output y_1 is always approximately equal to y_2 and the error between them is quite small, which indicates that the linear time-varying polynomial model can always capture the main features of the nonlinear six-DOF model.

5.2 MPC controller performance validation

In this section the performance of our adaptive MPC controller is tested. The data flow during our PIL simulation is shown in Fig. 10.

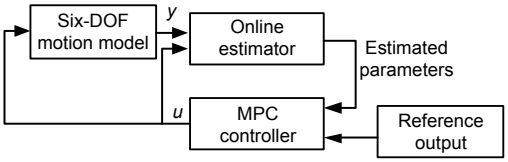

Fig. 10 Data flow of the processor-in-loop simulation

The six-DOF model operates in the host computer in MATLAB. It communicates with the target microcomputer through an RS232 cable. The online estimator and MPC controller work on the target board. The initial parameters for the online estimator are achieved in Table 4 in Section 5.1. Except for net buoyancy (Fig. 11), the initial condition of the six-DOF model is the same as in Section 5.1.

At first, we test the reference tracking performance when net buoyancy increases. The weights w_1 and w_2 are set as 1 and 0, respectively. Fig. 11 shows the time history of net buoyancy.

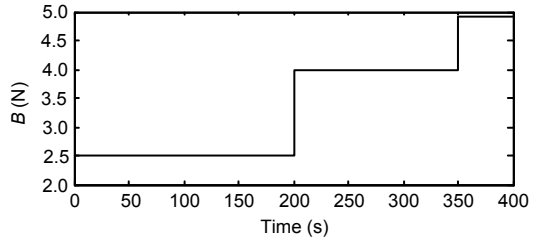

Fig. 11 Time history of net buoyancy during the processor-in-loop simulation

Two steps occur at 200 s and 350 s and the parameters estimated by the online estimator should respond to the steps. Fig. 12 shows the time history of parameters in vector ***A***. Figs. 13 and 14 show the zoom pictures of steps 1 and 2 in Fig. 12.

Fig. 15 shows the time history of parameters in vector ***B***.

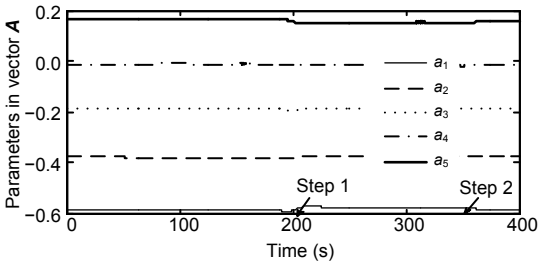

Fig. 12 Time history of parameters in vector *A* during processor-in-loop simulation

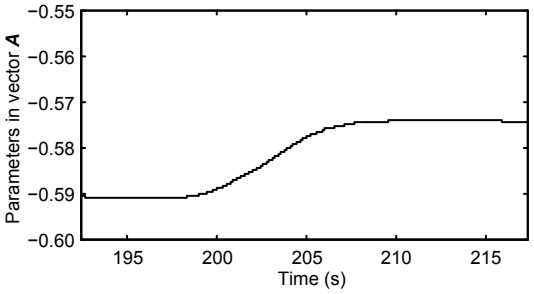

Fig. 13 View of step 1 in Fig. 12 (zoom in)

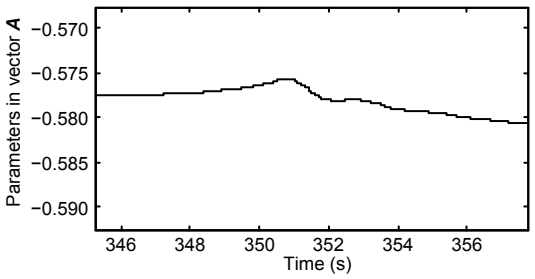

Fig. 14 View of step 2 in Fig. 12 (zoom in)

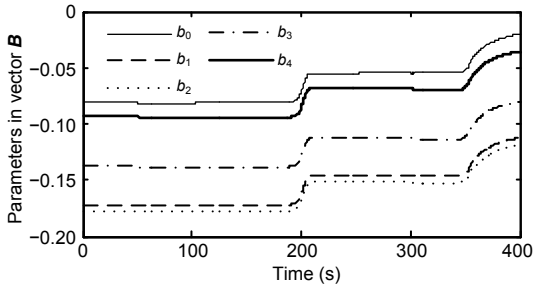

Fig. 15 Time history of parameters in vector *B*

Figs. 12 and 15 show that the online estimator can rapidly capture the changing dynamics features due to net buoyancy steps, and compute the adaptive parameters for the MPC controller to make the right decision. The controller performance is shown in Fig. 16.

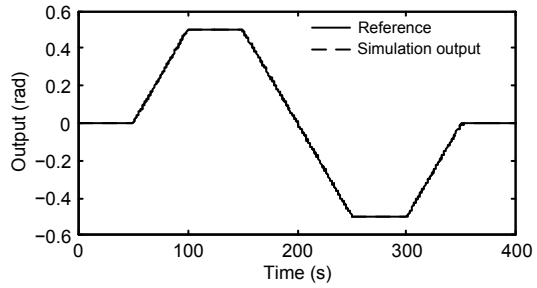

Fig. 16 Time history of processor-in-loop simulation output and reference output

Although the net buoyancy changes (Fig. 11) during the simulation, the adaptive MPC controller always works well. Fig. 16 shows the reference tracking performance during the simulation. As shown in this figure, the output is approximately the same as the reference.

Because w_2 is always set as 0, the above simulations ignore input move-suppression performance. Now we introduce how to tune w_2 to balance reference tracking and move suppression. Compared with move suppression, reference tracking is more important. Thus, w_1 should be larger than w_2. Increasing w_2 would be meaningful on the premise that w_1 is large enough to guarantee reference tracking performance. Figs. 17 and 18 exhibit the time history of inputs and outputs corresponding to w_2=0, 0.1, 0.2, 0.3, and 0.4. w_1 is always set as 1.

As shown in Fig. 17, the input suppression performance is improved (control command vibration becomes weak) as w_2 increases from 0 to 0.4. However, the output reference tracking shown in Fig. 18 becomes poor when w_2 is larger than 0.1. Thus, w_2=0.1 is a proper choice to balance both performances. As we know, severe input vibration can cause additional energy consumption to move inner

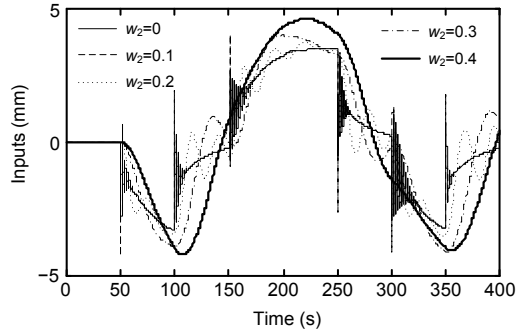

Fig. 17 Time history of inputs corresponding to different w_2

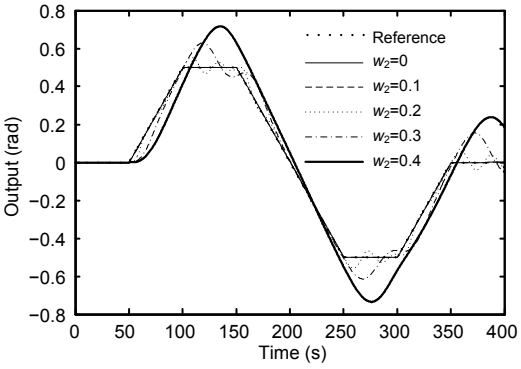

Fig. 18 Time history of outputs corresponding to different w_2

mass. Thus, input suppression can, at some level, represent energy-saving performance. Now we start to estimate the energy used to move inner mass corresponding to different w_2. Assuming that the underwater glider is static and that the CG location varies as shown in Fig. 17, we can use the following formulas to estimate the total work that the inner mass consumes for movement:

$$W_{\text{total}} = \sum_{i=1}^{3998} \left| F_i \cdot \Delta u_i \right|, \tag{44}$$

$$F_i = \frac{m}{m_l} \cdot m_l \cdot \frac{u_{i+2} - 2u_{i+1} + u_i}{0.01}, \tag{45}$$

$$\Delta u_i = \frac{m}{m_l} \cdot (u_{i+1} - u_i). \tag{46}$$

F_i represents the resultant force acted on the inner mass during the kth control interval. Δu_i represents the displacement during the kth control interval. m_l is the total mass of the inner movable mass and its mass is about 5 kg. m is the total mass of the glider. Thus, we have obtained the total work corresponding to different values of w_2 (Table 5). As shown in Table 5, it is concluded that increasing w_2 can improve the energy-saving performance of our steering controller.

Table 5 Consumed work corresponding to different w_2*

w_2	Work (J)
0	1.48
0.1	0.97
0.2	5.8×10^{-4}
0.3	2.9×10^{-6}
0.4	5.5×10^{-7}

* w_1=1

5.3 Polynomial structure validation

The polynomial structure used is based on Eq. (23), a fifth-order linear time-varying polynomial model with 10 parameters to be estimated. As we know, the more complicated polynomial structure we use, the more computational effort and memory we need. Compared to Eq. (23), Eq. (24) is less complicated and has only four parameters to be estimated. To confirm its control performance, we perform a PIL test and the results are presented in Figs. 19 and 20. Except for the polynomial structure, the simulation process here is the same as in Section 5.2.

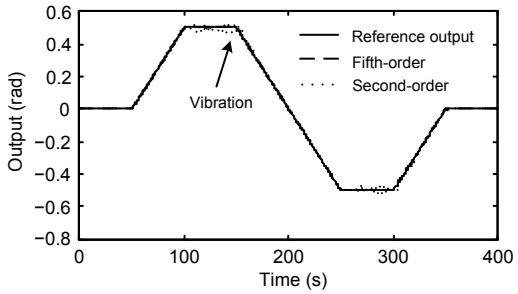

Fig. 19 Time history of outputs corresponding to different polynomial structures

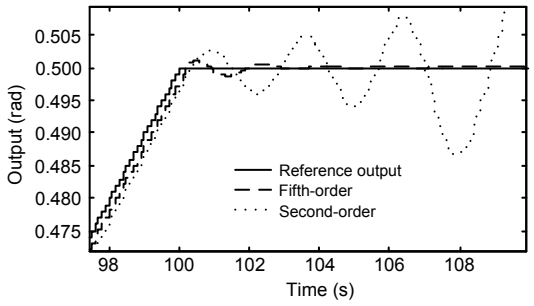

Fig. 20 View of the vibration zone in Fig. 19 (zoom in)

Fig. 19 shows that the reference tracking performance corresponding to the fifth-order polynomial structure is quite good, whereas the reference tracking performance corresponding to the second-order polynomial is poor and the output trembles significantly sometimes. Fig. 20 shows the partially enlarged vibration zone in Fig. 19.

5.4 Stability analysis

For a closed-loop control system, stability is the basic requirement. In this section, we verify this performance through monitoring the MPC controller's response to an unknown momentary disturbance. The disturbance is added to the yaw angle output during PIL simulation and it works on the host side. Fig. 21 shows the time history of disturbance.

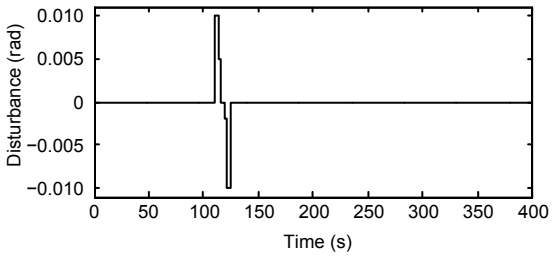

Fig. 21 Time history of the disturbance signal during processor-in-loop simulation

Two impulses occur at 110 s and 120 s. Both impulses occur for 5 s. Fig. 22 shows the closed-loop MPC control system response to the disturbance. Except for the added disturbance, the PIL simulation process here is the same as in Section 5.2.

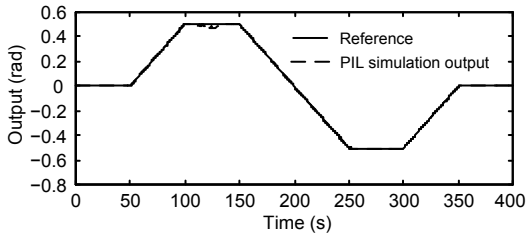

Fig. 22 Control system response to the disturbance signal during processor-in-loop simulation

As shown in Fig. 22, the yaw angle deviates from its reference when the impulse occurs at 110 to 125 s, and it gradually returns to its reference after the disturbance disappears. This PIL simulation indicates that our control system is stable and capable against a certain degree of disturbance.

6 In-lake experiments

In this section we introduce our in-lake experiments based on previous PIL simulations. Our glider is equipped with a digital compass (TCM5), a water pump, a servomotor, and a pressure sensor. The attitude angles (roll, pitch, and yaw angle) are recorded by the compass and filtered by a built-in low-pass FIR filter. The water pump can take in and expel water,

thereby changing the net buoyancy of the glider. The water is temporarily collected in a water tank when the glider dives and expelled when the glider surfaces. The capacity of the water tank is about 800 ml and the time rate of water flow can be measured by an electromagnetic flow meter. The total mass of the glider is about 49.193 kg with the water tank full. The servomotor is used to control the lateral CG location y_G by driving an inner movable mass and the mass weighs about 5 kg. Because the servomotor responds more quickly than yaw dynamics, the CG location is assumed to be the same as the control command computed by our MPC controller. The pressure sensor mentioned earlier is used to record the vertical depth of our glider in the lake. The CG location x_G is about 0.0013 m with the water tank full measured before the in-lake experiment. Our experiment consists of two steps:

1. Obtaining proper parameters for the controller to use in step 2. The net buoyancy remains constant with a value about 1 N measured before the in-lake experiment. The control command sent to the servomotor is shown in Fig. 23. During this step, the controller is always turned off. The online estimator works after 25 s to avoid the effects of waves (Fig. 24) when the glider is near the surface of the water. Figs. 25 and 26 show the time history of parameters recorded in step 1. Table 6 presents the values of the estimated parameters.

2. Testing the performance of the steering controller that we have designed when the net buoyancy increases. The flow rate remains steady at about 4 ml/s during our experiment and the net buoyancy is calculated through numerical integration for the flow rate recorded every 0.1 s. The steering controller and online estimator are turned down during the first 25 s of the experiment to avoid the effects of waves (Fig. 24) when the glider is near the surface of the water. After 25 s, the controller and online estimator start to work with the initial parameters obtained and shown in Table 6. Parameters w_1 and w_2 are set as 1 and 0.1, respectively.

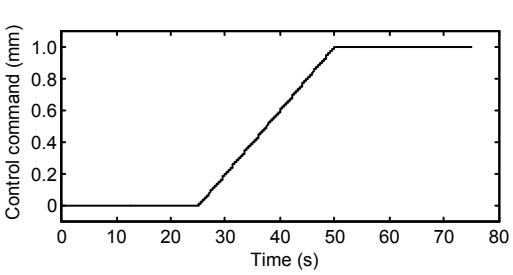

Fig. 23 Time history of the control command during the test

Fig. 24 The glider in the lake with waves

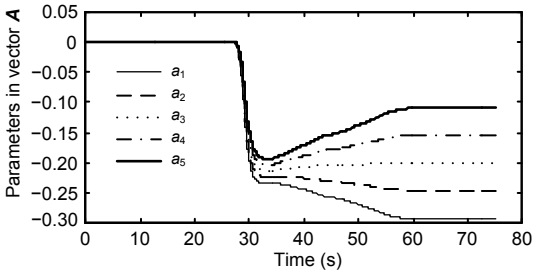

Fig. 25 Time history of parameters in vector *A* during the test

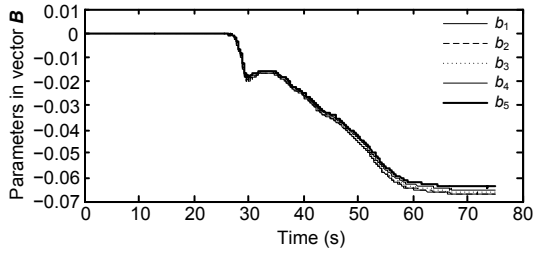

Fig. 26 Time history of parameters in vector *B* during the test

Table 6 Estimated parameters in vectors *A* and *B* during the test

Parameter	Value	Parameter	Value
a_1	−0.29	b_1	−0.067
a_2	−0.25	b_2	−0.067
a_3	−0.20	b_3	−0.066
a_4	−0.15	b_4	−0.065
a_5	−0.11	b_5	−0.064

Figs. 27–32 exhibit the time history of net buoyancy, depth, roll angle, pitch angle, yaw angle, and input (control command sent to the servomotor), respectively. Based on Fig. 31, we conclude that our controller works well in the experiment when net buoyancy increases (Fig. 27). As shown in Fig. 32, the fact that the input command trembles relatively slightly indicates that the input move-suppression performance meets our expectations.

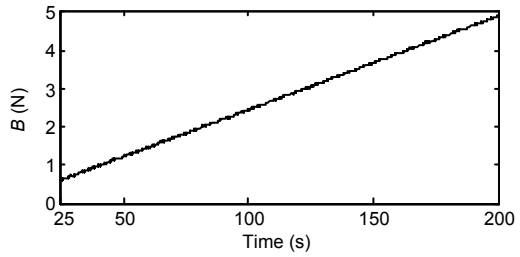

Fig. 27 Time history of net buoyancy during the test

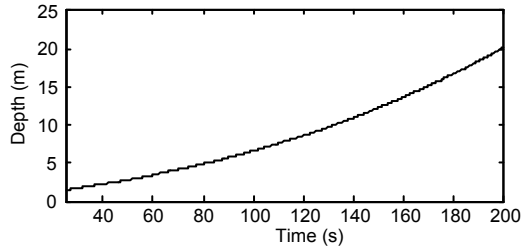

Fig. 28 Time history of depth during the test

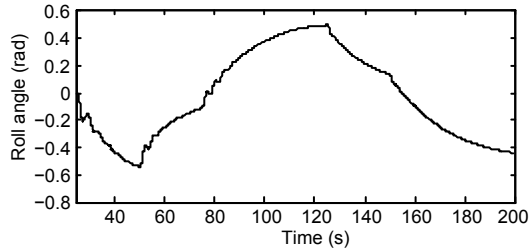

Fig. 29 Time history of the roll angle during the test

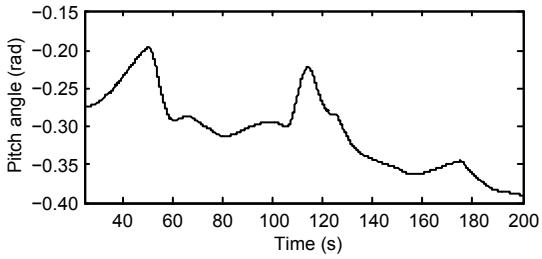

Fig. 30 Time history of the pitch angle during the test

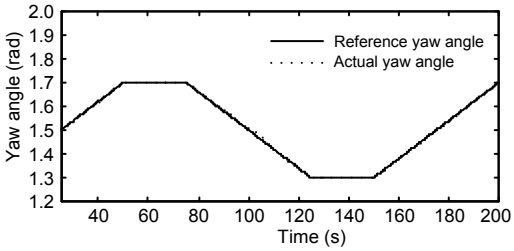

Fig. 31 Time history of the yaw angle during the test

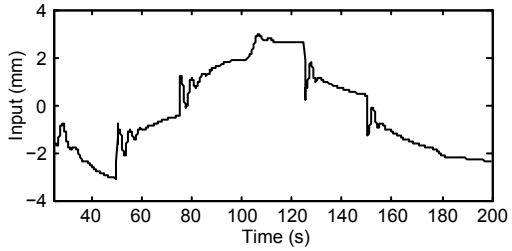

Fig. 32 Time history of the control command (y_G) during the test

7 Conclusions and perspective

We have proposed an adaptive, optimal, move-suppression (energy-saving) controller based on online system identification for underwater glider steering control. The online estimator can always capture the dynamic features of the six-DOF motion model when net buoyancy changes. Its convergence time is about 40 s and its prediction error is quite small compared with the six-DOF motion model output. It provides parameters for an adaptive controller to make optimal control decisions. The adaptive controller minimizes a cost function representing reference tracking and move suppression performance to compute the input command. We have tuned the weights to balance the competitive performances and concluded that w_1=1 and w_2=0.1 are proper choices for our MPC controller. We have compared the control performance using a second-order polynomial model to that using a fifth-order polynomial model in PIL simulations and found that the former cannot capture the main characteristics of yaw dynamics and may result in vibration during the flight. The PIL simulation can shorten the controller design cycle by detecting and solving potential problems before in-field experiments. The in-lake experiment results validate the performance of our steering controller when net buoyancy varies.

Because the controller ignores ocean current and surface wave effects, our future work is to take environmental disturbance into consideration, modify the algorithm if needed, and carry out ocean trials.

References

Alvarez, A., Caffaz, A., Caiti, A., *et al.*, 2009. Fòlaga: a low-cost autonomous underwater vehicle combining glider and AUV capabilities. *Ocean Eng.*, **36**(1):24-38. https://doi.org/10.1016/j.oceaneng.2008.08.014

Caccia, M., Indiveri, G., Veruggio, G., 2000. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *IEEE J. Ocean. Eng.*, **25**(2): 227-240. https://doi.org/10.1109/48.838986

Chen, H.T., 1981. Submarine Maneuverability. National Defense Industry Press, Beijing (in Chinese).

Eng, Y.H., Teo, K.M., Chitre, M., *et al.*, 2016. Online system identification of an autonomous underwater vehicle via in-field experiments. *IEEE J. Ocean. Eng.*, **41**(1):5-17. https://doi.org/10.1109/JOE.2015.2403576

Eriksen, C.C., Osse, T.J., Light, R.D., *et al.*, 2001. Seaglider: a long-range autonomous underwater vehicle for oceanographic research. *IEEE J. Ocean. Eng.*, **26**(4):424-436. https://doi.org/10.1109/48.972073

Graver, J.G., 2005. Underwater Gliders: Dynamics, Control and Design. PhD Thesis, Princeton University, New Jersey, USA.

Isa, K., Arshad, M.R., Ishak, S., 2014. A hybrid-driven underwater glider model, hydrodynamics estimation, and an analysis of the motion control. *Ocean Eng.*, **81**:111-129. https://doi.org/10.1016/j.oceaneng.2014.02.002

Leonard, N.E., Graver, J.G., 2001. Model-based feedback control of autonomous underwater gliders. *IEEE J. Ocean. Eng.*, **26**(4):633-645. https://doi.org/10.1109/48.972106

Li, J., Lu, C.J., Huang, X., 2010. Calculation of added mass of a vehicle running with cavity. *J. Hydrodynam. Ser. B*, **22**(3):312-318. https://doi.org/10.1016/S1001-6058(09)60060-3

Li, T.S., 1999. Torpedo Maneuverability. National Defense Industry Press, Beijing (in Chinese).

Liu, Y., Shen, Q., Ma, D.L., *et al.*, 2016. Theoretical and experimental study of anti-helical motion for underwater glider. *Appl. Ocean Res.*, **60**:121-140. https://doi.org/10.1016/j.apor.2016.09.001

Mišković, N., Vukić, Z., Bibuli, M., *et al*., 2011. Fast in-field identification of unmanned marine vehicles. *J. Field Robot.*, **28**(1):101-120. https://doi.org/10.1002/rob.20374

Phoemsapthawee, S., Le Boulluec, M., Laurens, J.M., *et al.*, 2011. Numerical study on hydrodynamic behavior of an underwater glider. Proc. 30th Int. Conf. on Ocean, Offshore and Arctic Engineering, p.521-526. https://doi.org/10.1115/OMAE2011-49670

Phoemsapthawee, S., Le Boulluec, M., Laurens, J.M., *et al*., 2013. A potential flow based flight simulator for an underwater glider. *J. Mar. Sci. Appl.*, **12**(1):112-121. https://doi.org/10.1007/s11804-013-1165-x

Rentschler, M.E., Hover, F.S., Chryssostomidis, C., 2006. System identification of open-loop maneuvers leads to improved AUV flight performance. *IEEE J. Ocean. Eng.*, **31**(1):200-208. https://doi.org/10.1109/JOE.2005.858369

Rudnick, D.L., Davis, R.E., Eriksen, C.C., *et al*., 2004. Underwater gliders for ocean research. *Mar. Technol. Soc. J.*, **38**(2):73-84. https://doi.org/10.4031/002533204787522703

Schmid, C., Biegler, L.T., 1994. Quadratic programming methods for reduced Hessian SQP. *Comput. Chem. Eng.*, **18**(9):817-832. https://doi.org/10.1016/0098-1354(94)E0001-4

Shi, S.D., 1995. Submarine Maneuverability. National Defense Industry Press, Beijing (in Chinese).

Tiano, A., Sutton, R., Lozowicki, A., *et al*., 2007. Observer Kalman filter identification of an autonomous underwater vehicle. *Contr. Eng. Pract.*, **15**(6):727-739. https://doi.org/10.1016/j.conengprac.2006.08.004

Wang, S.X., Sun, X.J., Wang, Y.H., *et al*., 2011. Dynamic modeling and motion simulation for a winged hybrid-driven underwater glider. *China Ocean Eng.*, **25**(1):97-112. https://doi.org/10.1007/s13344-011-0008-7

Xiao, Z.R., 2014. Calculation of submarine's inertial hydrodynamics coefficient. *J. Nav. Univ. Eng.*, **26**(5):44-47 (in Chinese).

Zhang, F.T., Tan, X.B., 2015. Passivity-based stabilization of underwater gliders with a control surface. *J. Dynam. Syst. Meas. Contr.*, **137**(6):1-13. https://doi.org/10.1115/1.4029078

Zhang, F.T., Zhang, F.M., Tan, X.B., 2014. Tail-enabled spiraling maneuver for gliding robotic fish. *J. Dynam. Syst. Meas. Contr.*, **136**(4):1-8. https://doi.org/10.1115/1.4026965