Excessive Disturbance Rejection Control of Autonomous Underwater Vehicle using Reinforcement Learning

Tianming Wang, Wenjie Lu, Dikai Liu
Centre for Autonomous Systems, University of Technology Sydney, Australia
tianming.wang@student.uts.edu.au, {wenjie.lu; dikai.liu}@uts.edu.au

Abstract

Small Autonomous Underwater Vehicles (AUV) in shallow water might not be stabilized well by feedback or model predictive control. This is because wave and current disturbances may frequently exceed AUV thrust capabilities and disturbance estimation and prediction models available are not sufficiently accurate. In contrast to classical model-free Reinforcement Learning (RL), this paper presents an improved RL for Excessive disturbance rejection Control (REC) that is able to learn and utilize disturbance behaviour, through formulating the disturbed AUV dynamics as a multi-order Markov chain. The unobserved disturbance behaviour is then encoded in the AUV state-action history of fixed length, its embeddings are learned within the policy optimization. The proposed REC is further enhanced by a base controller that is pre-trained on iterative Linear Quadratic Regulator (iLQR) solutions for a reduced AUV dynamic model, resulting in hybrid-REC. Numerical simulations on pose regulation tasks have demonstrated that REC significantly outperforms a canonical controller and classical RL, and that the hybrid-REC leads to more efficient and safer sampling and motion than REC.

1 Introduction

Large AUVs have been used in practical deep water applications, such as shipwreck search, underwater structure surveillance, and biology monitoring. In these applications, the strength and changes of external wave and current disturbances are negligible to the size of AUVs and thrust capabilities. In fact, hydrodynamic drag forces offer damping effects in favor of stabilizing the AUV systems. However, small AUVs in shallow water environments may be subject to disturbances whose magnitudes frequently exceed AUV thrust capacities, due to the turbulent flows and the AUV size limitations. These disturbances inevitably bring adverse effects and may even destabilize AUVs [Xie and Guo, 2000; Gao, 2014; Li *et al.*, 2014; Woolfrey *et al.*, 2016]. This paper studies the optimal control problem of a small AUV subject to unknown excessive disturbances, which may exceed its control capacities.

In the field of disturbance rejection control, feedback control strategies are used to suppress the unknown disturbances. Examples of feedback controllers include robust control [Skogestad and Postlethwaite, 2007], adaptive control [Åström and Wittenmark, 2013; Lu and Liu, 2017; 2018], optimal control [Bertsekas *et al.*, 1995], sliding mode control (SMC) [Edwards and Spurgeon, 1998], H-infinity control [Doyle *et al.*, 1989], etc. These methods often assume that the disturbance forces are within known bounds, which are usually smaller [Ghafarirad *et al.*, 2014] than control saturations, thus are unsuitable to this study.

One popular improvement to above approaches is to add a feedforward term based on the disturbance estimation [Yang *et al.*, 2010; Chen *et al.*, 2016]. Various disturbance estimation and attenuation methods have been proposed and practiced, such as disturbance observer (DOB) [Ohishi *et al.*, 1987; Chen *et al.*, 2000; Umeno *et al.*, 1993; Umeno and Hori, 1991], unknown input observer (UIO) in disturbance accommodation control (DAC) [Johnson, 1968; 1971], and extended state observer (ESO) [Han, 1995; Gao *et al.*, 2001]. However, such improvement on feedback controls is still unsuitable to this study, since disturbances exceed control bounds [Gao and Cai, 2016].

To this end, model predictive control (MPC) [Camacho and Alba, 2013] is often applied since it deals with constraints directly [Gao and Cai, 2016], through sacrificing instant performance for better overall performance during a fixed time horizon. MPC requires a sufficiently accurate prediction model of the robot system, and thus disturbance models built by DOB are used [Maeder and Morari, 2010; Yang *et al.*, 2010; 2011; Liu *et al.*, 2012; Yang *et al.*, 2014; Dirscherl *et al.*, 2015; Gao and Cai, 2016]. However, this model is quite difficult to obtain for the underwater robot subject to unknown varying disturbances [Maeder and Morari, 2010]. These disturbances are jointly determined by fluid conditions, robot morphologies, as well as varying robot states and controls. More importantly, such separated modeling and control optimization process might not be able to produce models and control signals that jointly optimize AUV performance, as evidenced in [Brahmbhatt and Hays, 2017; Karkus *et al.*, 2018].

This paper explores the integrated learning of disturbance behaviour and optimal controller through RL. RL is also known as adaptive dynamic programming and neural computing. Recently, deep RL algorithms based on Q-learning [Mnih *et al.*, 2015; Oh *et al.*, 2016; Gu *et al.*, 2016b], policy gradients [Schulman *et al.*, 2015a; Gu *et al.*, 2016a], and actor-critic methods [Lillicrap *et al.*, 2015; Mnih *et al.*, 2016; Schulman *et al.*, 2015b] have successfully solved problems in high-dimensional state spaces, where a system model is not available.

In modeling environmental behaviour, recurrent neural network has been used to model pedestrians' kinematics in [Alahi *et al.*, 2016], where the future pedestrians' trajectories are sufficiently embedded in pedestrians' current states. However, the future states of AUV do not only depend on the current states and actions, but also on the unknown disturbances, which are largely determined by turbulent flows with strong time correlation. Thus in this paper, we characterize the disturbed AUV dynamic system as a multi-order Markov chain. The unobserved disturbance behaviour is assumed to be encoded in the AUV state-action history of fixed length, its embeddings are learned within the policy optimization using Deep Deterministic Policy Gradient (DDPG) algorithm [Lillicrap *et al.*, 2015]. Therefore, in addition to the current states, the resultant trained policy also takes in a fixed length of state-action history to generate optimal control.

Model-free RL in general requires tremendous data that encodes the objective function and robot system dynamics (also known as transition model). While combining with some prior knowledge, such as a dynamic model or a controller, RL can significantly improve its sampling and thus learning efficiency. Kumar et al. [2018] and Koryakovskiy et al. [2018] both proposed to use model-free RL to learn a compensatory control signal on top of a model-based controller. The model-based controller can speed up learning of model-free RL and avoid risky exploratory actions, and the model-free learner can enhance the control performance by compensating the model-plant mismatch. However, the model-based controllers, such as MPC or LQR, may involve solving optimization problems, which is much slower than the forward propagation of a neural network policy. Nagabandi et al. [2018] also used model-based controller, but they used supervised learning to train an imitation policy to mimic the model-based controller, and then used this imitation policy as an initialization for the model-free learner.

In this study, the proposed REC is further enhanced by a base controller that is pre-trained on iLQR solutions for a reduced AUV dynamic model, resulting in hybrid-REC. The new actor network in hybrid-REC (also referred to as hybrid policy) is a summation of this fixed base controller and a trainable actor network same to REC. The latter one acts as a compensation term for the model-plant mismatch. The reduced AUV dynamic model does not consider wave and current disturbances. The iLQR is used to generate optimal controls and trajectories given random initial AUV states. Then supervised learning is used to train an imitation policy (a simple neural network) to mimic obtained optimal controls given any robot states as inputs. Afterwards, we use DDPG to train the new actor network and the critic network.

In this paper, Section 2 provides some preliminary knowledge about trajectory optimization and reinforcement learning. Section 3 introduces problem formulation. Section 4 and 5 provide the detailed description of REC and hybrid-REC algorithms. Then, Section 6 presents experimental validation procedures and result analysis.

2 Preliminaries

2.1 Trajectory Optimization

Trajectory optimization is the process of finding a state-control sequence which optimizes a given objective function [Tassa *et al.*, 2014]. Differential Dynamic Programming (DDP) is a second-order shooting method [Mayne, 1966] which under mild assumptions admits quadratic convergence for any system with smooth dynamics [Jacobson and Mayne, 1970]. Classic DDP requires second-order derivatives of the dynamics, which are usually the most expensive part of the computation. If only the first-order terms are kept, one obtains a Gauss-Newton approximation known as iterative Linear Quadratic Regulator (iLQR) [Li and Todorov, 2004; Todorov and Li, 2005], which is similar to Riccati iterations, but accounts for the regularization and line-search required to handle the nonlinearity.

We consider a system with discrete-time dynamics, but a similar derivation holds for the continuous case [Mayne, 1966]. The dynamics is modeled by a generic function f

$$s_{t+1} = f(s_t, a_t), \tag{1}$$

which describes the evolution from time t to $t+1$ of the state $s \in \mathcal{S} \in \mathbb{R}^n$, given the action $a \in \mathcal{A} \in \mathbb{R}^m$, where $\mathcal{S}$ and $\mathcal{A}$ represent state space and action space respectively. A trajectory $\{S, A\}$ is a sequence of controls $A = \{a_0, a_1, \cdots, a_{T-1}\}$, and corresponding state sequence $S = \{s_0, s_1, \cdots, s_T\}$ satisfying (1).

The total reward (the opposite number of cost) denoted by J is a sum of instant reward r and terminal reward r_f, incurred when the system starts from initial state s_0 and is controlled by the control sequence A until the horizon T is reached:

$$J(s_0, A) = \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) + \gamma^T r_f(s_T). \tag{2}$$

Indirect methods, like iLQR, represent the trajectory implicitly using only the controls A. The state sequence S is recovered by integration of (1) from the initial state s_0. The solution of the optimal control problem is the control sequence corresponding to the maximized total reward

$$A^\star = \arg\max_A J(s_0, A). \tag{3}$$

2.2 Reinforcement Learning

Model-free RL is a trial-and-error method that does not require an explicitly system model, and can naturally adapt to uncertainties in the real system [Sutton and Barto, 1998]. In RL, the goal is to learn a policy that chooses actions $a_t \in \mathcal{A}$ at each time step t in response to the current state $s_t \in \mathcal{S}$, such that the total expected sum of discounted rewards is maximized over all time. At each time step, the system transitions from s_t to s_{t+1} in response to the chosen action a_t and the transition dynamics function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, collecting a reward r_t according to the reward function $r(s_t, a_t)$. The discounted sum of future rewards is then defined as $\sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'} + \gamma^T r_f | s_t, a_t$, where $\gamma \in [0, 1)$ is a discount factor that prioritizes near-term rewards over distant rewards [Nagabandi *et al.*, 2018].

3 Problem Formulation

Our 6 degree of freedom (DOF) AUV is shown in Figure 1, the robot is designed to be sufficiently stable in roll and pitch even under strong disturbances, thanks to its large restoring forces. Thus, in order to simplify this problem, we only consider the control of the vehicle's position $p = [x \; y \; z]^T$ and yaw angle θ. The state of the robot s consists of the body position and yaw angle $q = [x \; y \; z \; \theta]^T \in \mathbb{R}^4$, as well as the corresponding velocities $\dot{q} \in \mathbb{R}^4$, then $s = [q^T \; \dot{q}^T]^T \in \mathbb{R}^8$. The action a includes the control forces and torques of the body $\tau_c \in \mathbb{R}^4$. The control limits are also taken into consideration $\underline{\tau}_{lim}, \overline{\tau}_{lim} \in \mathbb{R}^4$.

The robot model is simplified as a floating rigid body with external disturbances. The more detailed description of the dynamics function (1) for our robot system is given in the form:

$$M\ddot{q} + C\dot{q} + D\dot{q} + g = \tau_c + \tau_d, \tag{4}$$

$$\begin{bmatrix} q_{t+1} \\ \dot{q}_{t+1} \end{bmatrix} = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} + \begin{bmatrix} \dot{q}_t \\ \ddot{q}_t \end{bmatrix} dt, \tag{5}$$

where M is the inertia matrix, C is the matrix of Coriolis and centripetal terms, D is the matrix of drag force, g is the vector of the gravity and buoyancy forces, $\ddot{q}$ represents accelerations of the body, τ_d is the disturbance forces. In our case, we assume that the magnitudes of the disturbances are close to or exceed the robot control limits $\underline{\tau}_{lim}$ and $\overline{\tau}_{lim}$, but are constrained within a reasonable range, ensuring the controller is able to converge.

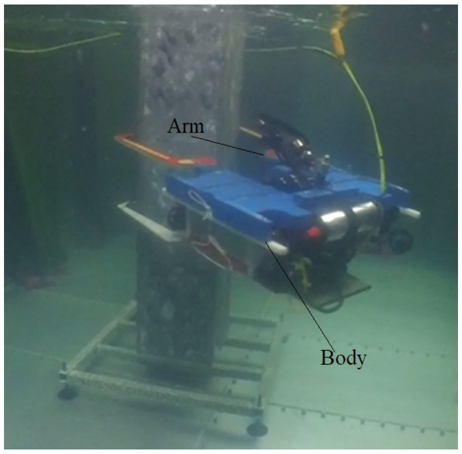


Figure 1: Submerged Pile Inspection Robot (SPIR) developed at Centre for Autonomous Systems (CAS), University of Technology Sydney (UTS)

4 REC Algorithm

The underwater disturbances mainly come from the time-varying current and wave, which have strong correlations in time. It means the disturbance behaviour can be learned for future disturbance prediction and thus for better control. Therefore, we characterize the disturbed AUV dynamic system as a multi-order Markov chain, and assume the unobserved varying disturbances and their predictions over next planning horizon are encoded in the AUV state-action history of fixed length $h_t = \{s_{t-H}, a_{t-H}, \cdots, s_{t-1}, a_{t-1}\}$, where H represents the length of the history. Thus the embeddings of the disturbance behaviour can be learned within the policy optimization. Note that actions are also included to encode disturbance behaviour in contrast to the state-only history, which, for example, has been used for estimating velocities in training video game player [Mnih *et al.*, 2015].

Before using this state-action history to train a policy, we first need to verify the rationality of the multi-order Markov chain hypothesis, through the validation of the existence of a dynamic model $s_{t+1} = f_h(h_t, s_t, a_t)$.

4.1 Validation of Multi-Order Markov Chain

In this part of work, a simple inverted pendulum model subject to external disturbance is used for validation. The learned dynamics function $\hat{f}_{h\theta}(h_t, s_t, a_t)$ is parameterized as a neural network, where the parameter θ represents the weights of the network. A straightforward parameterization for $\hat{f}_{h\theta}(h_t, s_t, a_t)$ would take the most recent history h_t, the current states s_t and actions a_t as inputs, and output the predicted next states $\hat{s}_{t+1}$. However, this function will be difficult to learn when the current states s_t and the next states s_{t+1} are too similar and the actions have little effect on the outputs; this difficulty becomes more pronounced as the time between states Δt becomes smaller and the state differences do not indicate the underlying dynamics well [Nagabandi *et* *al.*, 2018]. This issue is overcome by instead learning a dynamics function that predicts the change in states s_t over one time step duration Δt. Thus, the predicted next states are as follows: $\hat{s}_{t+1} = s_t + \hat{f}_{h\theta}(h_t, s_t, a_t)$.

Collecting Training Data: The training data is collected by sampling starting configurations $s_0 \sim P(s_0)$, generating random disturbance parameters, executing random actions at each time step, and recording the resulting trajectories $\tau = (s_0, a_0, \cdots, s_{T-1}, a_{T-1}, s_T)$ of length T.

Data Preprocessing: The trajectories $\{\tau\}$ are sliced into training data inputs (h_t, s_t, a_t) and corresponding output labels $s_{t+1} - s_t$. The useful training data should begin at $t = T - H$, since the agent starts to observe the full length of history at this time. The training data is then normalized and stored in the dataset $\mathcal{D}$.

Training Transition Model: The dynamic model $\hat{f}_{h\theta}(h_t, s_t, a_t)$ is trained by minimizing the error

$$\epsilon(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(h_t, s_t, a_t, s_{t+1}) \in \mathcal{D}} \frac{1}{2} ||(s_{t+1} - s_t) - \hat{f}_{h\theta}(h_t, s_t, a_t)||, \tag{6}$$

using stochastic gradient descent, where $|| \cdot ||$ represents L2-norm. While training on the training dataset $\mathcal{D}$, we also evaluate the loss in (6) on a evaluation dataset $\mathcal{D}_{eval}$, composed of trajectories not stored in the training dataset.

Through several experiments using the inverted pendulum model, we found that the error between the learned model and the actual model is always less than 2%, which proves the existence of the dynamic model and thus the rationality of the multi-order Markov chain hypothesis to some extent.

4.2 REC Architecture and Training

The rationality of the multi-order Markov chain hypothesis ensures that the REC algorithm is able to learn a satisfactory policy $\pi_\phi(a|h, s)$. In our implementations, DDPG [Lillicrap *et al.*, 2015] is used to train the neural netowrk policy. DDPG is an actor-critic, model-free algorithm based on the deterministic policy gradient that robustly solves challenging problems across a variety of domains with continuous action spaces. As shown in Figure 2, the REC algorithm consists of an actor network and a critic network. The actor network acts as a policy, which takes in the fixed length of state-action history as well as the current states to choose actions, the critic network is used to evaluate action-value function (discounted sum of future rewards) based on the state-action history, the current states and the selected actions. The action-value function and Temporal-Difference (TD) error are used respectively to update the parameters of the actor network and the critic network.

The algorithm details are shown in Algorithm 1. During training, our purpose is to enable the trained policy to deal with unknown varying disturbances, thus we randomly generate parameters of disturbances in each episode. Furthermore, in each episode, when the number of time steps does

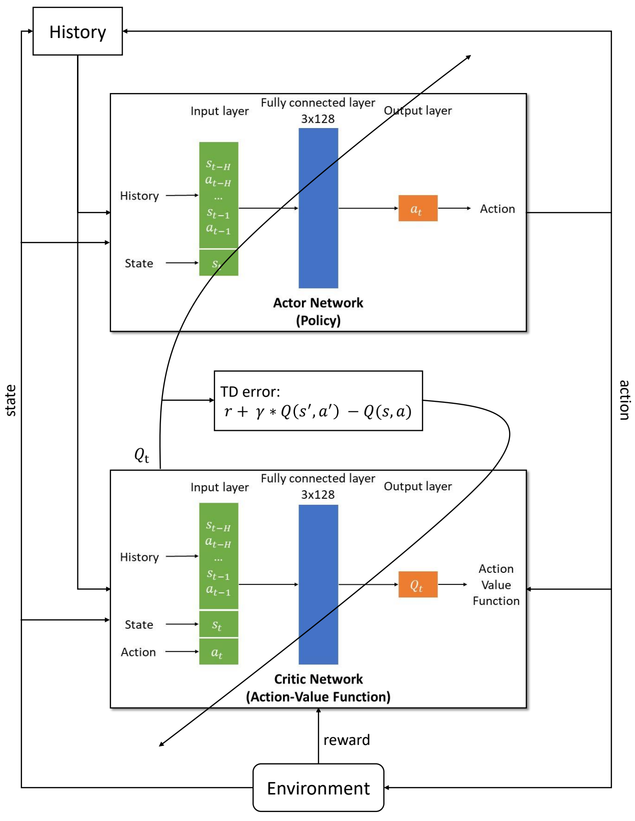


Figure 2: Network Architecture of REC

not reach the history length, the algorithm will randomly choose actions, and add current states and actions into the history. When the number of time steps exceeds the history length, the algorithm will choose actions based on the current deterministic policy, then update the history (delete the oldest state-action pair and add the latest one). The transition $(h_t, s_t, a_t, r_t, h_{t+1}, s_{t+1})$ for each step is saved to a replay memory. The training begins when the replay memory is full, a batch of N transitions is grabbed from the replay memory and used to train the actor and critic network at each step through minimizing actor loss funtion L_a and critic loss function L_c:

$$L_a = -\frac{1}{N} \sum_i Q(h_i, s_i, \pi(h_i, s_i)), \tag{7}$$

$$L_c = \frac{1}{N} \sum_i (y_i - Q(h_i, s_i, a_i))^2, \tag{8}$$

$$y_i = r_i + \gamma Q(h_{i+1}, s_{i+1}, \pi(h_{i+1}, s_{i+1})), \tag{9}$$

where y_i represents target action-value function. We also need to note that the disturbance behaviour is encoded in the state-ation history, thus during the training of the policy, the embeddings of the disturbance behaviour are also learned.

Algorithm 1: REC Algorithm

```
Randomly initialize critic network $Q(h, s, a)$ and
  actor network $\pi(h, s)$;
Initialize replay memory $R$;
for episode = 1, M do
    Receive initial observation state $s_0$;
    Initialize a random process $\mathcal{N}$ for action
      exploration;
    for t = 0, T-1 do
        if $t \leq$ history length then
            Select $a_t \in [\underline{\tau}_{lim}, \overline{\tau}_{lim}]$ randomly;
            Execute $a_t$ and observe $r_t$ and $s_{t+1}$;
            Add $s_t$ and $a_t$ into $h_{t+1}$;
        end
        else if $t >$ history length then
            Select $a_t \sim \pi(h_t, s_t) + \mathcal{N}_t$;
            Execute $a_t$ and observe $r_t$ and $s_{t+1}$;
            Update $h_t$ to $h_{t+1}$ by deleting $s_{t-H}$ and
              $a_{t-H}$ and adding $s_t$ and $a_t$;
            Store transition $(h_t; s_t; a_t; r_t; h_{t+1}; s_{t+1})$
              in $R$;
            if $R$ is full then
                Sample a random minibatch of $N$
                  transitions $(h_i; s_i; a_i; r_i; h_{i+1}; s_{i+1})$
                  from $R$;
                Update actor and critic by minimizing
                  the loss function (7) and (8);
            end
        end
        Update state: $s_t \leftarrow s_{t+1}$;
        Update history: $h_t \leftarrow h_{t+1}$;
    end
end
```

5 Hybrid REC Algorithm

Generic model-free RL in general requires tremendous data to converge to an optimal policy. While combining with some prior knowledge, such as a dynamic model or a controller, RL can significantly improve its sample efficiency. We propose a hybrid-REC algorithm for combining our REC algorithm with a base controller that is pre-trained on iLQR solutions for a reduced AUV dynamic model. The new actor network in hybrid-REC is a summation of this fixed base controller and the trainable actor network same to REC. The latter one acts as a compensation term for the outputs of base controller. The final control outputs are the combination of the base controller and the compensatory policy.

5.1 Base Controller

The base controller is obtained using iLQR [Li and Todorov, 2004; Todorov and Li, 2005]. The reduced dynamics functions are given by (4) and (5), excluding the disturbance term τ_d. Quadratic reward functions are used. We then optimize the sequence of actions $A = \{a_0, a_1, \cdots, a_{N-1}\}$ over a whole trajectory with length N through (3), using the given reduced dynamic model to predict future states.

Also, the control saturations need to be take into consideration when optimize the control sequence [Tassa *et al.*, 2014]. We consider control saturations of the form:

$$\underline{\tau}_{lim} \leq a \leq \overline{\tau}_{lim} \tag{10}$$

with element-wise inequality and $\underline{\tau}_{lim}$, $\overline{\tau}_{lim}$ the respective lower and upper bounds.

Trajectory optimizers are normally computationally expensive, since they need to solve an optimization problem every time they meet new initial states s_0, which make them not suitable for real-time operation. However, a neural network policy can calculate the control signals faster, the action selection only consumes the time for one forward propagation of the neural network. Thus, we then need to train a neural network policy to mimic our model-based controller.

The example trajectories are first gathered with the iLQR controller, which uses the given dynamics functions and the reward functions. The trajectories are collected into a dataset $\mathcal{D}^\star$, and then a neural network policy $\pi_\psi(a|s)$ is trained to match these expert trajectories in $\mathcal{D}^\star$. This policy's parameters are trained using the behavioral cloning objective [Nagabandi *et al.*, 2018]

$$\min_{\psi} \sum_{(s_t, a_t) \in \mathcal{D}^\star} ||a_t - \pi_\psi(s_t)||^2, \tag{11}$$

which we optimize using stochastic gradient descent.

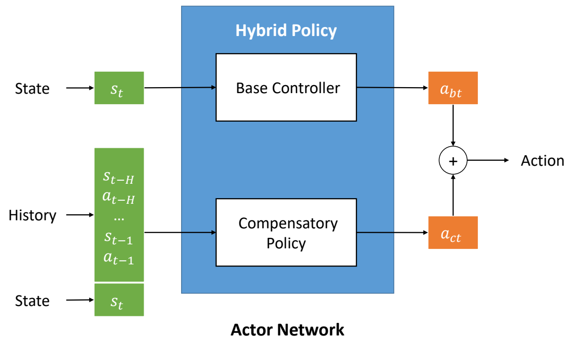


Figure 3: Actor Network Architecture of Hybrid REC

5.2 Hybrid REC Architecture and Training

Having the base controller trained on iLQR solutions, we then can build a parallel structure for the new actor network in the hybrid-REC algorithm (Figure 3). This network consists of the base controller with fixed parameters as well as a trainable neural network policy, which is used to compensate the outputs of the base controller in order for optimal control under external disturbances (also referred to as compensatory policy). The other parts of the network is same to REC.

The general training process is similar to Algorithm 1, except that two alternative policies are used for data sampling. In the beginning, the pre-trained base controller is used first for selecting actions and thus for data points sampling. After a certain number of episodes, the action selection policy is switched from the base controller to the hybrid policy. In the meantime, the training of the new actor-critic network is ongoing, this process is the same to REC, but only the compensatory term is trainable. The purpose of the switchable action selection policies is to avoid risky exploratory actions of the compensatory policy in the beginning, as the initial parameters of the neural network are randomly generated, leading to worse performance compared to the base controller in the beginning of policy optimization.

Some researchers [Nagabandi *et al.*, 2018] proposed to use the base controller as the initialization for the model-free RL. However, in order to deal with the disturbances, the model-free RL algorithm needs to use the state-action history along with the current states as the policy inputs, leading to different dimensions of input space for the base controller and the model-free policy. Thus the initialization of policy parameters is not feasible in our case.

6 Simulations

6.1 Simulation Setup

Our research addressed the control problems of an AUV subject to excessive external disturbances, we tested the performance of the proposed algorithms on pose regulation tasks. The robot has the mass $m = 60kg$ with the size of $0.8 \times 0.8 \times 0.25m^3$. The controls in roll and pitch of the robot are omitted, since the robot is designed to be sufficiently stable in roll and pitch even under strong disturbances, thanks to the large restoring forces. Thus, the robot has a 8-dimensional state space and a 4-dimensional action space. The control limits $\overline{\tau}_{lim} = -\underline{\tau}_{lim} = [120N\ 120N\ 80N\ 90Nm]^T$. In each episode of the experiment, the robot starts at a random pose, and it is controlled to reach a given pose and keep stable thereafter. The current disturbances are exerted on the x and y axes in the inertial frame.

In these experiments, we only consider disturbances in the form of sinusoidal waves with period ranging from 4s to 8s and phase ranging from 0 to 2π rad. Four different ranges of amplitude are provided, which are 50%-100%, 80%-120%, 100%-120% and 100%-150% of the robot control limits. Our purpose is to enable the trained policy to deal with unknown varying disturbance, thus the value of amplitude, period, and phase are randomly sampled from these distributions in each episode during training.

6.2 REC Results

The REC algorithm is applied to handle the disturbances through taking the state-action history and the current states as policy inputs. Different strength of the disturbances and different length of the history would affect the disturbance rejection performance. We first evaluate various disturbance amplitudes and history length for REC using empirical evaluations, we then compare REC with a canonical controller (RISE controller) and classical RL for their control performance subject to external disturbances.

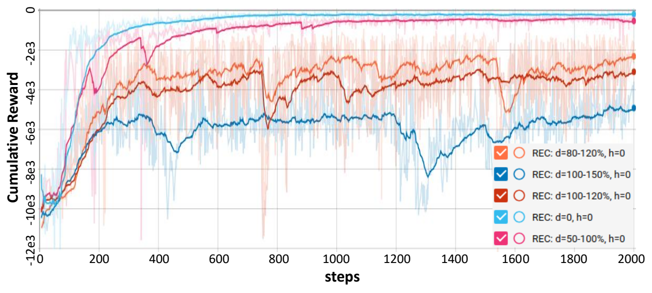


Figure 4: Comparison of different range of disturbance amplitudes for REC

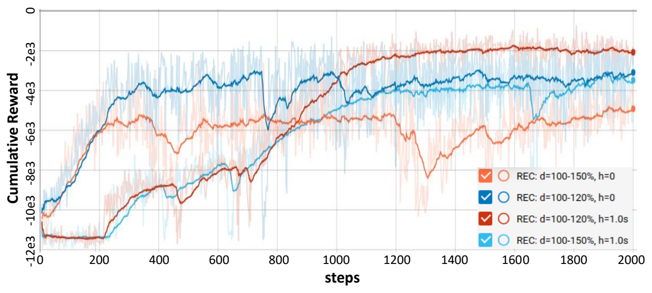


Figure 5: Comparison of using history or not for REC

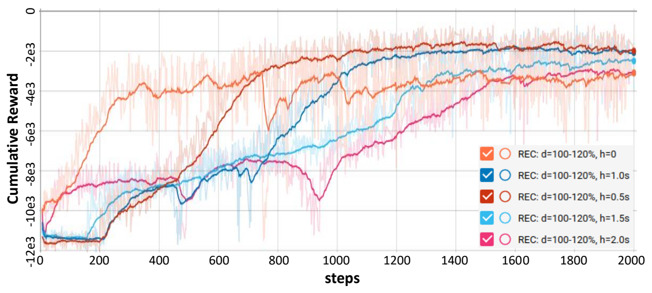
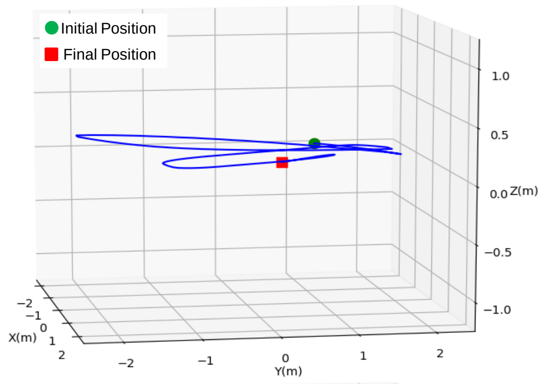


Figure 6: Comparison of different length of history for REC

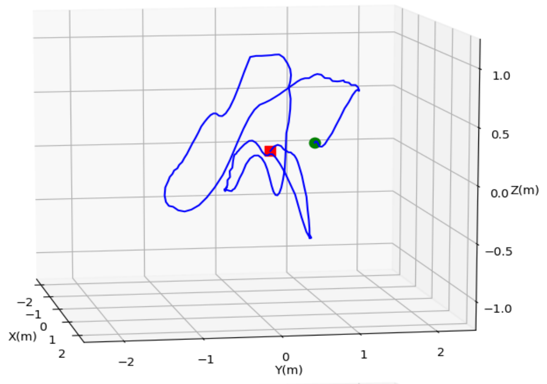
Figure 4 illustrates the training process of classical RL with five different ranges of disturbance amplitudes (including the situation without disturbances), showing that stronger disturbances lead to slower convergence speed and lower final cumulative reward, the results accord with our preconception. The figure also shows that, the performance won't be affected a lot if the disturbance amplitudes do not exceed the control limits (50%-100%). Once the disturbance amplitudes are larger than the control limits, the control performance decreases.

We used the situations that the disturbance amplitudes are

(a)



(b)



(c)

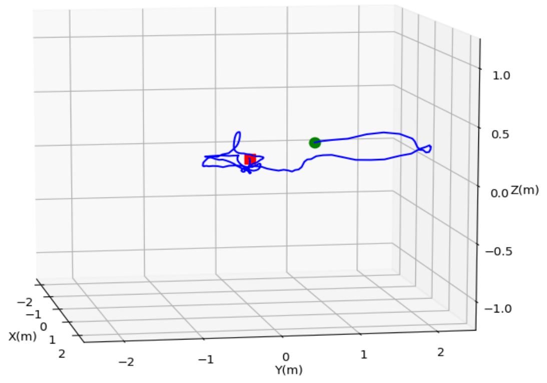


Figure 7: Comparison of 3D Trajectories: (a) RISE controller; (b) classical RL; (c) REC

larger than the control limits (100%-120% and 100%-150%) for further analysis. When taking 5-step history ($H = 5$) into consideration (as shown in Figure 5), both situations have better performance, which means the history information does improve the disturbance rejection capability. But the convergence speed apparently becomes slower, this might because the history information enlarges the state space, making the training process more difficult. In the following sections, we take the disturbance amplitudes of 100%-120% of the robot control limits as an example.

For the length of the history, Figure 6 shows that, using shorter history length gives faster convergence speed and better control performance, which means that the convergence speed is inversely proportional to the dimension of the state

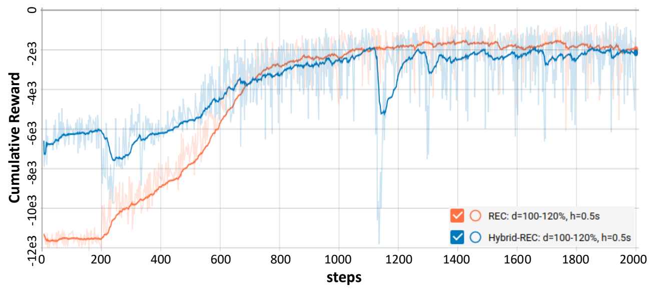


Figure 8: Comparison between REC and hybrid REC

space. However, we believe there should be an extremum for the control performance with respect to the history length, otherwise no history will be the best choice. This part of knowledge still requires further investigation.

We tested the control performance of RISE control [Fischer *et al.*, 2014], the classical RL and our REC algorithm, and recorded the 3D trajectories. As shown in Figure 7, given a random initial position ($X = 0.195m$, $Y = 0.861m$, $Z = 0.206m$) and a random set of disturbance parameters ($A_X = 130.168N$, $A_Y = 141.403N$, $T_X = 6.584s$, $T_Y = 7.855s$, $\phi_X = 0.438\pi$, $\phi_Y = 0.383\pi$), the robot is difficult to keep stable using either RISE controller or classical RL. While considering 5-step state-action history ($H = 5$) along with the current states as policy inputs (REC algorithm), the robot can quickly navigate to the target position and able to stabilize itself in a small range thereafter, which proves the effectiveness of our proposed algorithm.

6.3 Hybrid REC Results

We now compare the REC algorithm with the hybrid REC approach. Figure 8 shows that the hybrid REC starts with a higher cumulative reward (-6500 vs. -11500), but converges to an optimal value with nearly the same speed. This illustrates that the hybrid REC does avoid risky exploratory actions in the beginning, ensuring safer sampling and motion, but does not improve the sample efficiency significantly.

The reason for this phenomenon might be that, the base controller is trained for a reduced AUV model without disturbances, while the hybrid policy is trained using a disturbed AUV model. Figure 9 shows the state-action distribution in X axis, we can see that the base controller only has small control outputs when there is no disturbance. While the outputs of hybrid policy are mainly distributed around the control limits, except the region near the target. This might be because the impact of disturbances is quite strong, causing that the base controller cannot provide much help for the training process of the hybrid policy. The design parameters for the hybrid REC algorithm may also be a potential reason. As described in Section 5, the action selection policy is switched from the base controller to the hybrid policy after a certain number of training episodes. But how to choose the optimal number of episodes to switch policy remains unknown (cur-

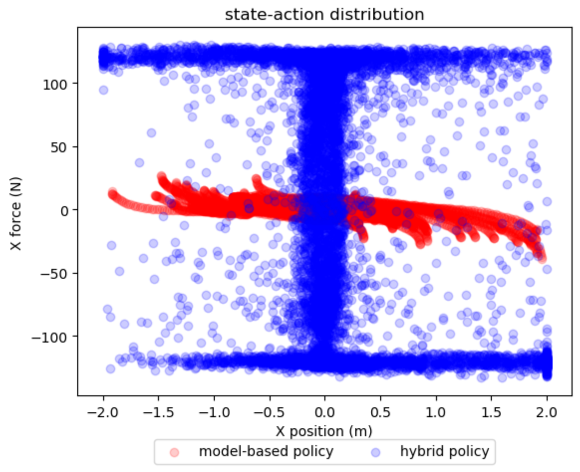


Figure 9: State Action Distribution

rent value is 200 episodes). We believe this part of work still requires further investigation.

7 Conclusion

In this paper, we presents an improved RL algorithm for excessive disturbance rejection control, REC. Through characterizing the disturbed AUV dynamic model as a multi-order Markov chain, the unobserved disturbance behaviour can be encoded in AUV state-action history of fixed length, and its embeddings can be learned with the policy optimization. A hybrid-REC algorithm has also been proposed to further improve the performance of REC, through combining a base controller that is pre-trained on iLQR solutions for a reduced AUV model, with a compensatory REC policy. Numerical simulations on pose regulation tasks have demonstrated that REC significantly outperforms RISE controller and classical RL, and that the hybrid-REC leads to more efficient and safer sampling and motion than REC.

While the effectiveness and simplicity of the hybrid REC algorithm is promising for ease of practical application, an interesting future work is to investigate the optimal combination of a base controller and compensatory policy, in order to further improve sampling efficiency. Another improvement is a better selection of the history length. The current algorithm directly takes a number of past states and actions as the policy inputs, this information could be utilized more sufficiently, for example, the Convolutional Neural Network (CNN) or Long Short Term Memory (LSTM) could be considered to deal with these history inputs. In addition, the deployment of this method on real-world robotic systems also requires future investigation, where the improved sample efficiency would make it practical to use even under the constraints of real-time sample collection in the real world.

Acknowledgments

This work was supported in part by the Australian Research Council (ARC) Linkage Project (LP150100935), the Roads and Maritime Services of NSW, and the Centre for Autonomous Systems (CAS) at the University of Technology Sydney.

References

[Alahi *et al.*, 2016] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[Åström and Wittenmark, 2013] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[Bertsekas *et al.*, 1995] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[Brahmbhatt and Hays, 2017] Samarth Brahmbhatt and James Hays. Deepnav: Learning to navigate large cities. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3087–3096, 2017.

[Camacho and Alba, 2013] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[Chen *et al.*, 2000] Wen-Hua Chen, Donald J Ballance, Peter J Gawthrop, and John O'Reilly. A nonlinear disturbance observer for robotic manipulators. *IEEE Transactions on industrial Electronics*, 47(4):932–938, 2000.

[Chen *et al.*, 2016] Wen-Hua Chen, Jun Yang, Lei Guo, and Shihua Li. Disturbance-observer-based control and related methodsan overview. *IEEE Transactions on Industrial Electronics*, 63(2):1083–1095, 2016.

[Dirscherl *et al.*, 2015] Christian Dirscherl, CM Hackl, and Korbinian Schechner. Explicit model predictive control with disturbance observer for grid-connected voltage source power converters. In *Industrial Technology (ICIT), 2015 IEEE International Conference on*, pages 999–1006. IEEE, 2015.

[Doyle *et al.*, 1989] John C Doyle, Keith Glover, Pramod P Khargonekar, and Bruce A Francis. State-space solutions to standard h/sub 2/and h/sub infinity/control problems. *IEEE Transactions on Automatic control*, 34(8):831–847, 1989.

[Edwards and Spurgeon, 1998] Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications*. Crc Press, 1998.

[Fischer *et al.*, 2014] Nicholas Fischer, Devin Hughes, Patrick Walters, Eric M Schwartz, and Warren E Dixon. Nonlinear rise-based control of an autonomous underwater vehicle. *IEEE Transactions on Robotics*, 30(4):845–852, 2014.

[Gao and Cai, 2016] Haiyan Gao and Yuanli Cai. Nonlinear disturbance observer-based model predictive control for a generic hypersonic vehicle. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 230(1):3–12, 2016.

[Gao *et al.*, 2001] Zhiqiang Gao, Yi Huang, and Jingqing Han. An alternative paradigm for control system design. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 5, pages 4578–4585. IEEE, 2001.

[Gao, 2014] Zhiqiang Gao. On the centrality of disturbance rejection in automatic control. *ISA transactions*, 53(4):850–857, 2014.

[Ghafarirad *et al.*, 2014] Hamed Ghafarirad, Seyed Mehdi Rezaei, Mohammad Zareinejad, and Ahmed AD Sarhan. Disturbance rejection-based robust control for micropositioning of piezoelectric actuators. *Comptes Rendus Mécanique*, 342(1):32–45, 2014.

[Gu *et al.*, 2016a] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[Gu *et al.*, 2016b] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[Han, 1995] Jingqing Han. The" extended state observer" of a class of uncertain systems [j]. *Control and Decision*, 1, 1995.

[Jacobson and Mayne, 1970] David H Jacobson and David Q Mayne. Differential dynamic programming. 1970.

[Johnson, 1968] C Johnson. Optimal control of the linear regulator with constant disturbances. *IEEE Transactions on Automatic Control*, 13(4):416–421, 1968.

[Johnson, 1971] Cn Johnson. Accomodation of external disturbances in linear regulator and servomechanism problems. *IEEE Transactions on automatic control*, 16(6):635–644, 1971.

[Karkus *et al.*, 2018] Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks: End-to-end probabilistic localization from visual observations. *arXiv preprint arXiv:1805.08975*, 2018.

[Koryakovskiy *et al.*, 2018] Ivan Koryakovskiy, Manuel Kudruss, Heike Vallery, Robert Babuška, and Wouter Caarls. Model-plant mismatch compensation using reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):2471–2477, 2018.

[Kumar *et al.*, 2018] Visak CV Kumar, Sehoon Ha, and Katsu Yamane. Improving model-based balance controllers using reinforcement learning and adaptive sampling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7541–7547. IEEE, 2018.

[Li and Todorov, 2004] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[Li *et al.*, 2014] Shihua Li, Jun Yang, Wen-Hua Chen, and Xisong Chen. *Disturbance observer-based control: methods and applications*. CRC press, 2014.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Liu *et al.*, 2012] Cunjia Liu, Wen-Hua Chen, and John Andrews. Tracking control of small-scale helicopters using explicit nonlinear mpc augmented with disturbance observers. *Control Engineering Practice*, 20(3):258–268, 2012.

[Lu and Liu, 2017] Wenjie Lu and Dikai Liu. Active task design in adaptive control of redundant robotic systems. In *Australasian Conference on Robotics and Automation*. ARAA, 2017.

[Lu and Liu, 2018] Wenjie Lu and Dikai Liu. A frequency-limited adaptive controller for underwater vehicle-manipulator systems under large wave disturbances. In *The World Congress on Intelligent Control and Automation*, 2018.

[Maeder and Morari, 2010] Urban Maeder and Manfred Morari. Offset-free reference tracking with model predictive control. *Automatica*, 46(9):1469–1476, 2010.

[Mayne, 1966] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim

Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Nagabandi *et al.*, 2018] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, pages 7579–7586. IEEE, 2018.

[Oh *et al.*, 2016] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *arXiv preprint arXiv:1605.09128*, 2016.

[Ohishi *et al.*, 1987] Kiyoshi Ohishi, Masato Nakao, Kouhei Ohnishi, and Kunio Miyachi. Microprocessor-controlled dc motor for load-insensitive position servo system. *IEEE Transactions on Industrial Electronics*, (1):44–49, 1987.

[Schulman *et al.*, 2015a] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[Schulman *et al.*, 2015b] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[Skogestad and Postlethwaite, 2007] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Tassa *et al.*, 2014] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1168–1175. IEEE, 2014.

[Todorov and Li, 2005] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306. IEEE, 2005.

[Umeno and Hori, 1991] Takaji Umeno and Yoichi Hori. Robust speed control of dc servomotors using modern two degrees-of-freedom controller design. *IEEE Transactions on Industrial Electronics*, 38(5):363–368, 1991.

[Umeno *et al.*, 1993] Takaji Umeno, Tomoaki Kaneko, and Yoichi Hori. Robust servosystem design with two degrees of freedom and its application to novel motion control of robot manipulators. *IEEE Transactions on Industrial Electronics*, 40(5):473–485, 1993.

[Woolfrey *et al.*, 2016] Jonathan Woolfrey, Dikai Liu, and Marc Carmichael. Kinematic control of an autonomous underwater vehicle-manipulator system (auvms) using autoregressive prediction of vehicle motion and model predictive control. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4591–4596. IEEE, 2016.

[Xie and Guo, 2000] Liang-Liang Xie and Lei Guo. How much uncertainty can be dealt with by feedback? *IEEE Transactions on Automatic Control*, 45(12):2203–2217, 2000.

[Yang *et al.*, 2010] Jun Yang, Shihua Li, Xisong Chen, and Qi Li. Disturbance rejection of ball mill grinding circuits using dob and mpc. *Powder Technology*, 198(2):219–228, 2010.

[Yang *et al.*, 2011] Jun Yang, Shihua Li, Xisong Chen, and Qi Li. Disturbance rejection of dead-time processes using disturbance observer and model predictive control. *Chemical engineering research and design*, 89(2):125–135, 2011.

[Yang *et al.*, 2014] Jun Yang, Zhenhua Zhao, Shihua Li, and Wei Xing Zheng. Nonlinear disturbance observer enhanced predictive control for airbreathing hypersonic vehicles. In *Control Conference (CCC), 2014 33rd Chinese*, pages 3668–3673. IEEE, 2014.