DINAME 2019 - Proceedings of the XVIII International Symposium on Dynamic Problems of Mechanics,
M.A. Savi, T.G. Ritto and W.M. Bessa (Editors), ABCM, Buzios, RJ, Brazil, March 10th to 15th, 2019

Feedback Linearization and Supervisioned Neural Networks for the Depth Control of a ROV

Diago Cesar Xavier de Freitas Barros[1], **Gabriel da Silva Lima**[1], **Wallace Moreira Bessa**[1]

[1] RoboTeAM - Robotics and Machine Learning, Department of Mechanical Engineering, Federal University of Rio Grande do Norte, Natal, Brazil.

Abstract: A ROV (Remotely Operated underwater Vehicle) system needs a precise estimation of its position in order to avoid damage and imprecise movements in certain missions. In this paper, the intelligent control of the path trajectory of a ROV susceptible to external forces (water dynamics, currents, animals, etc.) is proposed based on feedback linearization and use of an error estimator with neural network. This work proposes an approach to the matter using a series of different neural networks, under supervised learning, for the unknown dynamics, to evaluate the performance of each network case, both in time and error prediction between the ideal trajectory and the real due to disturbances.

Keywords: ROV, intelligent control, feedback linearization, neural networks.

INTRODUCTION

ROVs are being used for underwater exploration, in the research of subsea phenomena and in the assembly, inspection and repair of offshores structures, due to substitute human operators inside the vehicle in tasks that may result in risk to human life.

The execution of a certain tasks with the robotic vehicle, the operator needs to monitor and control a certain number of parameters. If some of these parameters, as for instance the position and attitude of the vehicle, could be controlled automatically, the teleoperation of the ROV can be enormously facilitated. (Bessa, Dutra and Kreuzer, 2008).

The dynamic positioning of unmanned underwater vehicles has been a recurrent study in the newest papers and confirms the necessity of the development of a controller that could deal with the inherent nonlinear system dynamics, imprecise hydrodynamic coefficients, and external disturbances (Bessa, Dutra and Kreuzer, 2010).

In the case of underwater vehicles, the traditional control methodologies are not the most suitable choice and cannot guarantee the desired performance and precision (Goheen and Jefferys, 1990 and Yuh, 1994).

The control system is one of the most important elements of an underwater robotic vehicle, and its characteristics (advantages and disadvantages) play an essential role when one has to choose a vehicle for a specific mission (Bessa, Dutra and Kreuzer, 2010). The strategy to control the position of a ROV can be done using a Feedback Linearization (FL), sliding modes (SMC), Proportional Integrative and Derivative (PID), etc. It is important to create a robust and efficient controller that can predict the movement and compensate the disturbances plausible to happen on the movement of the ROV, once certain missions requires a high precision on its position to accomplish some desired goal.

A FL is an approach used to control non-linear systems, as it involves the transformation of the non-linear system into an equivalent system that is linear, using only a change of variables and suitable control input. One model of FL is the input-output linearization, which objective is to linearize the map of the transformed inputs, by the change of variables, and the actual output model. This is an important approach to a non-linear problem, since requires less equation to control and its precise and robust enough if well implemented, once the dynamic model can be effectively known.

In this paper, a FL is proposed to compensate the dynamics of the vertical movement of the ROV subject to a constant hydrodynamic force and noise in the motion of the system. An Artificial Neural Network (ANN) updates the parameters of this controller, to make the control more effective and robust as possible, once we present the controller the model of the dynamics of the system. Figure 1 represent a block diagram of the presented control system.

The update parameters occurred in two different neural network, the Rosenblatt's perceptron and Radial Basis Function (RBF) in order to achieve the best network to update the parameters in the lowest time and with the greatest precision. The learning algorithm used to train the networks was based on the gradient descent and pseudoinverse.

The algorithms used to make the adjustments are essential to this paper, since its construction are the point to be evaluated, as the controller is being actualized by the output of the ANN in sort of way to accelerate the convergence of the system. Moreover it's described how they were developed to make this ROV system stable and working in the required area of error state.

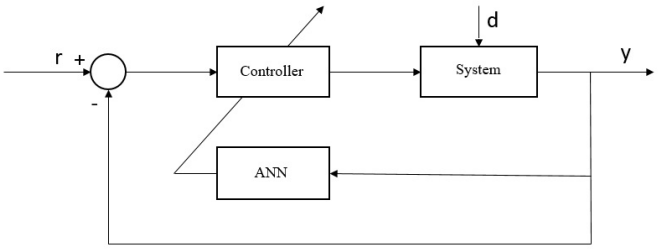


Figure 1: Proposed control strategy.

CONTROL LAW

An adequate model to describe the dynamic behavior of an underwater robotic vehicle must include the rigid-body dynamics of the vehicle and a representation of the surrounding fluid dynamics (Bessa, Dutra and Kreuzer, 2008). The equations of motion for underwater vehicles can be expressed, with respect to the body-fixed reference frame, in the following vectorial form:

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{k}(\boldsymbol{v}) + \boldsymbol{h}(\boldsymbol{v}) + \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{u} \tag{1}$$

where $\boldsymbol{v} = [v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^\top$ is the vector of linear and angular velocities in the body-fixed reference frame, $\boldsymbol{x} = [x\ y\ z\ \alpha\ \beta\ \gamma]^\top$ represents the position and orientation with respect to the inertial reference frame, $\boldsymbol{M}$ is the inertia matrix, $\boldsymbol{k}(\boldsymbol{v})$ is the vector of generalized Coriolis and centrifugal forces, $\boldsymbol{h}(\boldsymbol{v})$ represents the hydrodynamic effects, $\boldsymbol{g}(\boldsymbol{x})$) is the vector of generalized restoring forces (gravity and buoyancy), and $\boldsymbol{u}$ is the vector of control forces and moments.

It should be noted that in the case of a ROV, the metacentric height is sufficiently large to provide the self-stabilization of roll (α) and pitch (β) angles. This allows the order of the dynamic model to be reduced to four degrees of freedom, $\boldsymbol{x} = [x\ y\ z\ \gamma]^\top$, and the vertical motion to be decoupled from the motion in the horizontal plane. Thus, the positioning system of a ROV can be divided in two different parts: depth control (concerning variable z) and control in the horizontal plane (variables x, y and γ) (Bessa, Dutra and Kreuzer, 2008).

The vertical motion of a ROV is expressed, considering the assumptions above and that the buoyance condition is neutral therefore only the hydrodynamic drag is opposed to the corpse movement (Bessa, Dutra and Kreuzer, 2010), by:

$$m\ddot{z} = -c\dot{z}|\dot{z}| + u \tag{2}$$

The change of variables for the system above is required to operate the problem as a linear system. The strategy of FL proposed, so that the system is guaranteed the stability, is presented in the Eq. 3, which the vertical acceleration is calculated by a difference between the control signal and hydrodynamics forces. Equation 4, shows that the control signal are dependent of a λ, a variable strictly positive, $\tilde{z}$, the tracking error and a compensator for non-modelled dynamic $\hat{d}$, provided by the neural network.

$$\ddot{z} = m^{-1}(u - d) \tag{3}$$

$$u = m(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z}) + \hat{d} \tag{4}$$

If the estimate error $\tilde{d} = d - \hat{d}$ is bounded, i.e, $|\tilde{d}| \leq \varepsilon$, so it is can possible to proof that the control law (4) converges the error to a limited region $E = \{(\tilde{z}, \dot{\tilde{z}}) \in \mathbb{R}^2 \mid |\tilde{z}^{(i)}| \leq (i+1)!\lambda^{i-2}\varepsilon, i = 0, 1\}$ (Tanaka, Fernandes and Bessa, 2013).

ARTIFICIAL NEURAL NETWORK

ANN is a mathematical strategy to approximate function, estimate values, approximation function and it model is inspired by the human neural function. An ANN usually includes neurons, connections and biases. Neurons are arranged in layers (Mohammadzaheri, Chen, and Grainger, 2012).

In control applied with ANN, it is a difficult problem to find the best ANN structure for each specific problem; thus, a certain amount of ANN is usually employed to deal with relatively complex approximation problems, to find the best network suited for the task (Deng, Li, and Wu, 2008).

Rosenblatt's Perceptron

In Rosenblatt's Perceptron, the summing node of the neural model computes a linear combination of the inputs applied to its synapses, as well as incorporates an externally applied bias. The resulting sum is then applied to a hard limiter

function to evaluate this sum and, if pass, the output (Gallant, 1990). Equation (5) defines this network mathematically.

$$y = \sum_{i=1}^{n} w_i x_i + b \tag{5}$$

where y is the output vector, w a weights for each neuron, x the inputs vector and b a bias.

The Rosenblatt's Perceptron is a simple model, but with a mathematical refinement can be used to solve problems like the position tracking of the ROV with a great performance. The most important part is to make a adjustment for the algorithm to solve the problem proposed, with the amount of change in the implementation, can cause a subtle change that enhances the overall potential of the model. This ANN receives the position, velocity and a bias as input, to estimate the disturbance signal d as a factor $\hat{d}$ to be provided in the model for improve the control. The proposed algorithm is presented in Appendix (Alg. 1).

Radial Basis Function

The RBF are functions that has a different mathematical approach for the convergence of the output of the network. Essentially, the most important difference in the usage of a RBF, as proposed below, resides in the function regression for the output reach the goal of update the parameters correctly and as a result make the controller reach a steady-state point more fast and in a stable way for a system that require fast responses.

The RBF typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. Equation (6) demonstrates mathematically the RBF, for an error dynamic $\sigma = \dot{\tilde{z}} + \lambda \tilde{z}$ and a Gaussian activation function φ.

$$y = \sum_{i=1}^{n} w_i \varphi_i(\sigma) \tag{6}$$

The learning algorithms of the networks are applied to train the network in a as fast and precise way as possible. The gradient descent is a well-known method to actualize the weights of the neural networks in a iterative optimization process, as Eq. 7 shows the new weights calculated in each time step using this method, where the η is the learning rate and $\frac{\partial e}{\partial \boldsymbol{w}}$ is the quadratic error. The proposed algorithm is presented in Appendix (Alg. 2).

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \eta \frac{\partial e}{\partial \boldsymbol{w}} \tag{7}$$

The second method of RBF consist in the pseudoinverse matrix of the space states of the system, using a convergence method described in Deep Learning (Haykin, 2007), in which the matrix has some parameters that allow always this pseudoinverse to exist.

As for the pseudoinverse technique is represented mathematically in Eq. 8, where there is a matrix $\boldsymbol{G}^+$, composed by a product between the pseudo inverse matrix and her transpose, that multiply the expected response d and then update the weights, considering the regularization parameter ϕ is close enough to zero. The proposed algorithm is presented in Appendix (Alg. 3).

$$\boldsymbol{w} = \boldsymbol{G}^+ d \tag{8}$$

The $\boldsymbol{G}^+$ matrix can be computed with the following expression:

$$\boldsymbol{G}^+ = (\boldsymbol{G}^\top \boldsymbol{G} + \phi \boldsymbol{G}_c)^{-1} \boldsymbol{G}^\top \tag{9}$$

where $\boldsymbol{G} = \boldsymbol{G}(\sigma)$ is a matrix composed by Green functions φ (Haykin, 2007):

$$\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{c}) = \begin{bmatrix} \varphi(x_1, c_1) & \varphi(x_1, c_2) & \dots & \varphi(x_1, c_n) \\ \varphi(x_2, c_1) & \varphi(x_2, c_2) & \dots & \varphi(x_2, c_n) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(x_n, c_1) & \varphi(x_n, c_2) & \dots & \varphi(x_n, c_n) \end{bmatrix} \tag{10}$$

In this work, the Green function $\varphi(\sigma, c_i) = \varphi_i(\sigma)$ is given by the quadratic exponential function:

$$\varphi_i(\sigma) = \exp\left(-\frac{(\sigma - c_i)^2}{2\gamma^2}\right) \tag{11}$$

where c_i is the i-center of neuron i and γ is the length-scale. Graphically, these activation functions can be viewed in the Fig. 2.

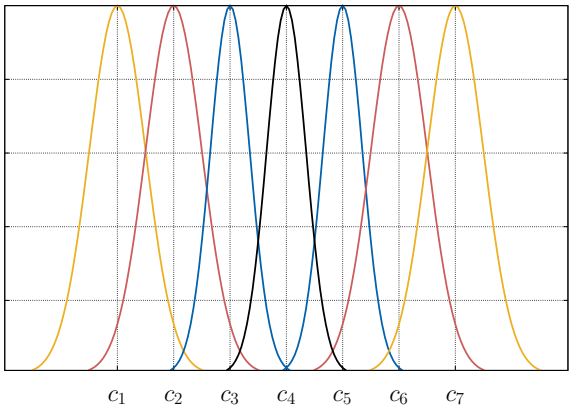
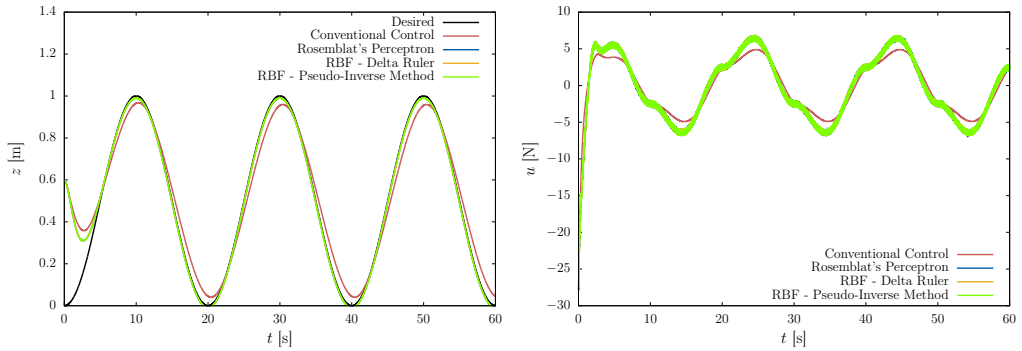


Figure 2: RBF activation functions.

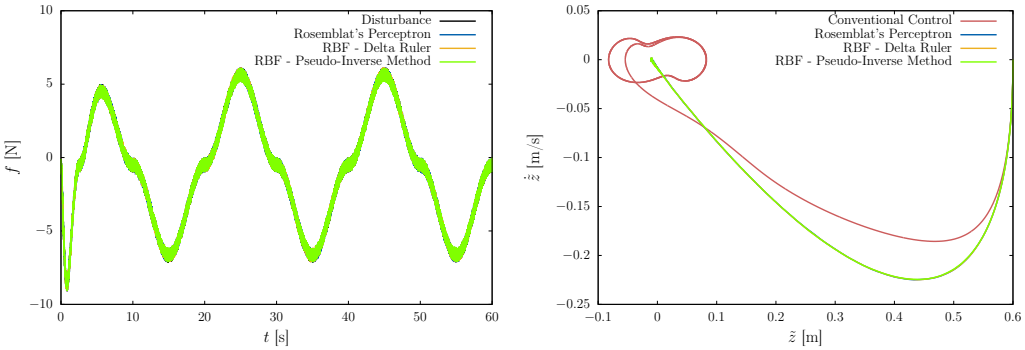
NUMERICAL RESULTS

The proposed control scheme is now evaluated by means of numerical simulations. The fourth order Runge-Kutta method is employed and sampling rates of 200 Hz for the controller and 500 Hz for system dynamics are assumed. In the dynamic model (Eq. 2) was considered that $m = 50$ kg and $c = 250$ kg/m. Guassian noise with $\sigma = 0.02$ is added to the $\ddot{z}$ in order to emulate noisy signal. And the controller parameter was chosen $\lambda = 1$. The desired trajectory is given by $z_d = 0.5\,(1 - \cos\,(2\pi t/20))$. The simulation results are presented in the Fig. 3.



(a) Trajectory tracking.

(b) Control signal.



(c) Disturbance.

(d) Phase portrait with FL + ANN.

Figure 3: Trajectory tracking with conventional (FL) and proposed (FL + ANN) control approaches.

After training the neural network, results of the position show that the classical control strategy has failed in track the desired path of the ROV as for the FL with the active networks showed an almost perfect approximation of the vertical system trajectory. With the addition of a compensator disturbance, the control signal increase in relation to the case without compensator.

The improved performance of the proposed controller is due to ANN ability to estimate and compensate for unstructured uncertainties, Fig. 3c.

The efficacy of ANN can be clearly ascertained by comparing the phase portrait of the tracking error obtained with both conventional and proposed control strategies, respectively (Fig. 3d). Without proper compensation, modeling uncertainties may lead to limit cycles in the closed-loopsystem, as observed in Fig. 3d. The adoption of a ANN compensator within the feedback linearization controller proved to be quite effective in tracking the desired trajectory, even considering a large initial tracking error.

All the ANN tested showed similar results, which implies that the supervisioned learning estimates the disturbance with great precision. The mean squared was calculated for the control signal and position error and for each method used. They are presented in the Tab. 1. As it can be seen, the ANN compesator increase the mean squared in comparison with conventional control. On the other hand, the square mean error decrease in relation to the conventional control.

Table 1: Mean squared.

Variable	Conventional Control	Rosemblat's Perceptron	RBF - Delta Ruler	RBF - Pseudo-Inverse
Control Signal (u)	13.7273200	20.7434139	20.7395211	20.6879134
Position Error ($\tilde{z}$)	0.0109357	0.0074528	0.0074494	0.0074569

CONCLUSIONS

In this paper, a feedback linearization controller is combined with three different types of supervisioned learning to ANN to deal with the trajectory tracking of a ROV. By means of numerical simulations, the improved performance of the proposed scheme over the conventional feedback linearization controller is demonstrated.

It's shown that, for this specific problem, with the data used for the constant buoyancy force and disturbance, all the algorithms worked exceptionally compared to convectional control, but with a very low discrepancies between their results. Perhaps a problem with more complexity would show more robustness of one over another.

As a suggestion for future works, are proposed to use the algorithms for other practical problems, to specify their effectiveness between themselves.

For the problem proposed, it's seen that a simple Rosenblatt's Percetron can solve with great performance the problem, a ideal scenario since its a very simple and not a time demanding algorithm to compile, which show a promise to work onboard in real machine problems, in which are necessary rapid calculations for the controller actuate before a catastrophe occur.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES).

REFERENCES

Bessa, W.M., Dutra, M.S. and Kreuzer, E., 2008, Depth control of remotely operated underwater vehicles using an adaptive fuzzy sliding mode controller, Robotics and Autonomous Systems, Vol.56, pp. 670-677.

Bessa, W.M., Dutra, M.S. and Kreuzer, E., 2010, An adaptive fuzzy sliding mode controller for remotely operated underwater vehicles, Robotics and Autonomous Systems, Vol.58, pp. 16-26.

Bessa, W.M., Dutra, M.S. and Kreuzer, E., 2013, Dynamic Positioning of Underwater Robotic Vehicles with Thruster Dynamics Compensation, International Journal of Advanced Robotic Systems, Vol.10.

J. Yuh, J., 1994. Learning control for underwater robotic vehicles, IEEE Control Systems Magazine, Vol. 14, No. 2, pp. 39-46.

K.R. Goheen, E.R. Jeffreys, 1990. Multivariable self-tuning autopilots for autonomous and remotely operated underwater vehicles, IEEE Journal of Oceanic Engineering, Vol. 15 ,No. 3, pp. 144-151.

Mohammadzaheri, M., Chen, L. and Grainger, S., 2012. A Critical Review of the most Popular Types of Neuro Control. Asian Journal of Control, Vol. 11, pp. 1-11.

Deng, H., H.-X. Li, and Y.-H. Wu, 2008. Feedback linearization-based neural adaptive control for unknown nonaffine nonlinear discrete-time systems, IEEE Trans. Neural Netw., Vol. 19, No. 9, pp. 1615-1625.

Gallant, S. I., 1990. Perceptron-based learning algorithms. IEEE Transactions on Neural Networks, Vol. 1, No. 2, pp. 179-191.

Haykin, S., 2007, Redes Neurais Princípios e Prática, Bookman 2^a ed., 898 p.

Tanaka, M.C., Fernandes, J.M.M., Bessa, W.M., 2013. "Feedback Linearization with Fuzzy Compensation for Uncertain Nonlinear Systems", International Journal of Computers Communications & Control, Vol. 8, No. 5, pp. 736-743.

RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.

APPENDIX

Algorithm 1: Feedback Linearization Control and Rosenblatt's Perceptron.

Define control parameters
Define initial states: $z_0, \dot{z}_0$
Define initial weights and bias: $\boldsymbol{w}_0 = [b_0, w_{1_0}, w_{2_0}]^\top$
while $t \leq t_{fim}$ **do**
  Evaluate desired trajectory: $\boldsymbol{z}_d$
  Compute tracking error: $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}_d$
  $\hat{d} \leftarrow \boldsymbol{w}_i^\top \boldsymbol{z}_i$
  $u \leftarrow m(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z}) + \hat{d}$
  Compute the dynamic estimate error: $f \leftarrow u - m\ddot{z}$
  Apply u in the dynamic model
  Update the states: $z, \dot{z}$
  Compute the dynamic estimate error: $e \leftarrow \frac{1}{2}(f - \boldsymbol{w}_i^\top \boldsymbol{z}_i)^2$
  while $e > 10^{-3}$ **do**
    $\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \eta(f - \boldsymbol{w}_i^\top \boldsymbol{z}_i)\boldsymbol{z}_i$
    Compute the dynamic estimate error again: $e \leftarrow \frac{1}{2}(f - \boldsymbol{w}_{i+1}^\top \boldsymbol{z}_i)^2$
  end
  $t \leftarrow t + \Delta t$
end

Algorithm 2: Feedback Linearization Control and RBF with gradient descent.

Define control parameters
Define initial states: $z_0, \dot{z}_0$
Define initial weights and bias: $\boldsymbol{w}_0$
while $t \leq t_{fim}$ **do**
  Evaluate desired trajectory: $\boldsymbol{z}_d$
  Compute tracking error: $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}_d$
  $\sigma_i = \dot{\tilde{z}} + \lambda\tilde{z}$
  $\hat{d} \leftarrow \boldsymbol{w}_i^\top \boldsymbol{\varphi}(\sigma_i)$
  $u \leftarrow m(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z}) + \hat{d}$
  Compute the dynamic estimate error: $f \leftarrow u - m\ddot{z}$
  Apply u in the dynamic model
  Update the states: $z, \dot{z}$
  $\sigma_{i+1} = \dot{\tilde{z}} + \lambda\tilde{z}$
  Compute the dynamic estimate error: $e \leftarrow \frac{1}{2}(f - \boldsymbol{w}_i^\top \boldsymbol{\varphi}(\sigma_{i+1}))^2$
  while $e > 10^{-3}$ **do**
    $\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \eta(f - \boldsymbol{w}_i^\top \boldsymbol{\varphi}(\sigma_{i+1}))\boldsymbol{\varphi}(\sigma_{i+1})$
    Compute the dynamic estimate error again: $e \leftarrow \frac{1}{2}(f - \boldsymbol{w}_{i+1}^\top \boldsymbol{\varphi}(\sigma_{i+1}))^2$
  end
  $t \leftarrow t + \Delta t$
end

Algorithm 3: Feedback Linearization Control and RBF with pseudoinverse technique.

Define control parameters
Define initial states: $z_0, \dot{z}_0$
Define initial weights and bias: $\boldsymbol{w}_0$
while $t \leq t_{fim}$ **do**
    Evaluate desired trajectory: $\boldsymbol{z}_d$
    Compute tracking error: $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}_d$
    $\sigma_i = \dot{\tilde{z}} + \lambda \tilde{z}$
    $\hat{d} \leftarrow \boldsymbol{w}_i^\top \boldsymbol{\varphi}(\sigma_i)$
    $u \leftarrow m(\ddot{z}_d - 2\lambda \dot{\tilde{z}} - \lambda^2 \tilde{z}) + \hat{d}$
    Compute the dynamic estimate error: $f \leftarrow u - m\ddot{z}$
    Apply u in the dynamic model
    Update the states: $z, \dot{z}$
    $\sigma_{i+1} = \dot{\tilde{z}} + \lambda \tilde{z}$
    $\boldsymbol{G}^+ = (\boldsymbol{G}^\top \boldsymbol{G} + \phi \boldsymbol{G}_c)^{-1} \boldsymbol{G}^\top$
    $\boldsymbol{w}_{i+1} = \boldsymbol{G}^+ f$
    $t \leftarrow t + \Delta t$
end