2015-05

Robust Adaptive Control of an Uninhabited Surface Vehicle

Annamalai, ASK

10.1007/s10846-014-0057-2
Journal of Intelligent & Robotic Systems
Springer Science and Business Media LLC

Journal of Intelligent and Robotic Systems

Robust adaptive control of an uninhabited surface vehicle

--Manuscript Draft--

Manuscript Number:	
Full Title:	Robust adaptive control of an uninhabited surface vehicle
Article Type:	Full/Regular paper
Keywords:	Uninhabited surface vehicles, gradient descent, least squares, weighted least squares, adaptive control, robust control
Corresponding Author:	Andy SK Annamalai School of Marine Science and Engineering, Plymouth University Plymouth, UNITED KINGDOM
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	School of Marine Science and Engineering, Plymouth University
Corresponding Author's Secondary Institution:	
First Author:	Andy SK Annamalai
First Author Secondary Information:	
Order of Authors:	Andy SK Annamalai
	Robert Sutton
	Chenguang Yang
	Phil Culverhouse
	Sanjay Sharma
Order of Authors Secondary Information:	
Abstract:	A robust adaptive autopilot for uninhabited surface vehicles (USV) based on a model predictive controller (MPC) is presented in this paper. The novel autopilot is capable of handling sudden changes in system dynamics. In real life situations, very often a sudden change in dynamics results in missions being aborted and the uninhabited vehicles have to be rescued before they cause damage to other marine craft in the vicinity. This problem has been suitably dealt with by this innovative design. The MPC adopts an online adaptive nature by utilising three algorithms, individually: gradient descent, least squares and weighted least squares (WLS). Even with random initialisation, significant improvements over the other algorithmic approach were achieved by WLS by maintaining the intermittent continuous values of system parameters and periodically reinitilaising the covariance matrix. Also, a time frame of 25 seconds appears to be the optimum to reinitialise the parameters. This novel approach enables the autopilot to cope well with significant changes in the system dynamics and empowers USVs to accomplish their desired missions.

Robust adaptive control of an uninhabited surface vehicle

A SK Annamalai[1] • R Sutton[1] • C Yang[2] • P Culverhouse[2] • S Sharma[1]
[1](*School of Marine Science and Engineering, Plymouth University, Plymouth, UK*)
[2](*School of Computing and Mathematics, Plymouth University, Plymouth, UK*)
(E-mail: andy.annamalai@plymouth.ac.uk)

Abstract

A robust adaptive autopilot for uninhabited surface vehicles (USV) based on a model predictive controller (MPC) is presented in this paper. The novel autopilot is capable of handling sudden changes in system dynamics. In real life situations, very often a sudden change in dynamics results in missions being aborted and the uninhabited vehicles have to be rescued before they cause damage to other marine craft in the vicinity. This problem has been suitably dealt with by this innovative design. The MPC adopts an online adaptive nature by utilising three algorithms, individually: gradient descent, least squares and weighted least squares (WLS). Even with random initialisation, significant improvements over the other algorithmic approach were achieved by WLS by maintaining the intermittent continuous values of system parameters and periodically reinitilaising the covariance matrix. Also, a time frame of 25 seconds appears to be the optimum to reinitialise the parameters. This novel approach enables the autopilot to cope well with significant changes in the system dynamics and empowers USVs to accomplish their desired missions.

Keywords

Uninhabited surface vehicles, gradient descent, least squares, weighted least squares, adaptive control, robust control

Classification code

93 Systems theory; control

1 Introduction

For a number of years *Predator* unmanned air vehicles have been involved in strike missions using air-to-ground *Hellfire* missiles. However, surprisingly, the first missiles launched by the US Navy from an uninhabited surface vehicle (also known as unmanned surface vehicle) (USV) took place during trials in late October 2012 [1]. During these trials in total six Rafael *Spike* missiles were fired which equates to a total payload displacement of approximately 204 kg. Clearly if a pod of such missiles was to be launched in a salvo then there would be an abrupt change in the dynamic characteristics of the vehicle owing to the sudden decrease in its overall mass. Whereas, the mass of the vehicle would gradually alter over a period of time should the missiles be discharged individually at a spasmodic rate.

Besides coping with changing payloads, as illustrated above, multi-role USVs also have to contend with amendments to mission requirements and objectives, and varying environmental conditions whilst being employed in the commercial, naval and scientific sectors. Hence all USVs have a common need for robust adaptive control (autopilot) systems. Thus, to date, in order to meet the testing demands being imposed by these various sectors, autopilots have been designed based on fuzzy [2], gain scheduling [3], H infinity [4] linear quadratic Gaussian [5], sliding mode [6], neural network [7] and local control network [8] techniques that have met with varying degrees of success.

Since management and monitoring of the environment is a major issue worldwide, an USV named *Springer*, depicted in Fig 1, has been specifically built and continues to be developed to be a cost effective and environmentally friendly vehicle primarily for undertaking pollutant tracking, and environmental and hydrographical surveys in rivers, reservoirs, inland waterways and coastal waters, particularly where shallow waters prevail. An equally important secondary role is also envisaged for *Springer* as a test bed platform for other academic and scientific institutions involved in environmental data gathering, sensor and instrumentation technology, control systems engineering and power systems based on alternative energy sources.

Fig 1 *Springer* USV

Thus for the reasons outlined above this paper reports a study into the application of gradient descent, least squares and weighted least squares in-conjunction with MPC techniques in the design of adaptive autopilots for the *Springer* vehicle. In particular the study investigates the capabilities of the autopilots to cope with a sudden change in the mass of the USV.

With regards to the structure and content of the paper, on completion of this introductory material, section 2 outlines the *Springer* vehicle hardware, and presents the yaw dynamic model used in simulation studies. Whilst section 3 describes the autopilot designs, and in section 4 results and discussion are presented. Finally concluding remarks are given in section 5.

2 The *Springer* uninhabited surface vehicle

As full details of the *Springer's* hardware have already been reported in [9], only an outline will be presented here. The *Springer* USV was designed as a medium waterplane twin hull vessel which is versatile in terms of mission profile and payload. It is 4.2m long and 2.3m wide with a displacement of 0.6 tonnes. Each hull is divided into three watertight compartments. The sensor and computer systems are carried in watertight Peli cases. This facilitates the quick substitution of systems on shore or at sea. In order to prevent any catastrophe resulting from a water leakage, leak sensors are utilized within the motor housing. A mast has also been installed to carry the GPS and wireless antennas. The wireless antenna is used as a means of communication between the vessel and its user and is intended to be utilized for remote monitoring purposes, intervention in the case of erratic behaviour and to alter the mission parameters.

The *Springer* propulsion system consists of two propellers powered by a set of 24V 74lbs (334N) Minn Kota Riptide transom mounted saltwater trolling motors. As will be seen below in Fig 2, steering of the vessel is based on differential propeller revolution rates. The vehicle has a differential steering mechanism and thus required two inputs to adjust its course. This was simply modelled as a two input, single output system in the form depicted in Fig 2.

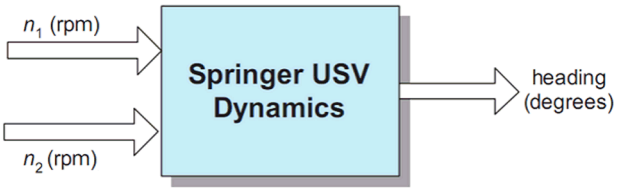


Fig 2 Block diagram representation of *Springer* USV

where n_1 and n_2 being the two propeller thrusts in revolutions per minute (rpm). Clearly, straight line manoeuvres require both the thrusters running at the same speed whereas the differential thrust is zero in this case. By letting n_c and n_d represent the common mode and differential mode thruster velocities defined then they are defined by

$$n_c = \frac{n_1 + n_2}{2} \tag{1}$$

$$n_d = \frac{n_1 - n_2}{2} \tag{2}$$

In order to maintain the velocity of the vessel, n_c must remain constant at all times. The differential mode input, however, oscillates about zero depending on the direction of the manoeuvre. Please note that whilst the actual steerage system operates using rpm.

Furthermore, the block diagram of the autopilot and the USV are shown in the following Fig 3. The guidance system based on line of sight generates the reference angle whereas the autopilot keeps the vehicle on-course. Since it is based on MPC the actuator limitations are inherently taken into account. The output of the USV is compared against the reference and the error generated is further utilised to generate a better control action.

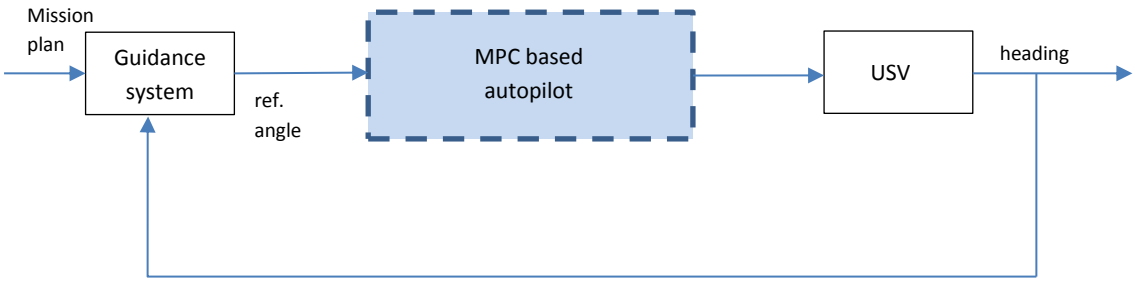


Fig 3 Autopilot of *Springer* USV

The dynamic model of the *Springer* vehicle was obtained in state space form as shown in equation (3).

$$\begin{aligned} \boldsymbol{x}(k+1) &= \mathbf{A}\boldsymbol{x}(k) + \mathbf{B}u(k) \\ y(k) &= \mathbf{C}\boldsymbol{x}(k) + \mathbf{D}u(k) \end{aligned} \tag{3}$$

Where

$$A = \begin{bmatrix} 0.0372 & 1 & 0 & 0 \\ 0.1002 & 0 & 1 & 0 \\ 0.3781 & 0 & 0 & 1 \\ 0.471 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0.3235 \\ -0.8218 \\ 0.8819 \\ -0.3031 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \qquad D = \begin{bmatrix} 0 \end{bmatrix} \tag{4}$$

This model was obtained from offline data and does not reflect the current true dynamics of the system at all times during its operation. Environmental changes, wear and tear and mass changes all attribute to the change in system dynamics and can offset the autopilot performance if it is based on an offline model obtained from prior trials / experiments. Hence to improve the overall performance, it was contemplated to update the model of the plant at each sampling instant. The next section illustrates how this is achieved and implemented on *Springer* USV.

3 Autopilot Designs

The autopilot is concerned with keeping the vehicle on course. In this research, the MPC has been utilised as an autopilot, as it offers considerable benefits by enforcing various types of constraints on the vehicle. The concepts and developments of MPC, over the past three decades are covered in [10].

Authors such as in [11], [12], [13], [14] and [15], epitomise the recent adaption of MPC in marine control system design. The techniques of MPC have been well established in the process and petrochemical industries for eons as illustrated by [16], [17], [18], [19].

MPC employs a model of the vessel internally to predict the output (as shown in Fig 4a). The accuracy of the model determines the efficiency of the controller and the controller effort required to keep the vehicle on course with a minimum effort [20]. When there is a sudden change in the plant dynamics, employing a static internal model would have a fatal flaw in the autopilot design and severely undermine the success of the missions undertaken by the USV. To overcome this problem, adaptive algorithms as mentioned previously were employed to construct an internal model of the vessel (which reflects the vessels current dynamics as accurately as possible) at any given point in time.

This model was utilised to generate predicted output against set reference trajectories. A cost function of the following form is used to define how well the predicted vessel output was able to track the set reference point.

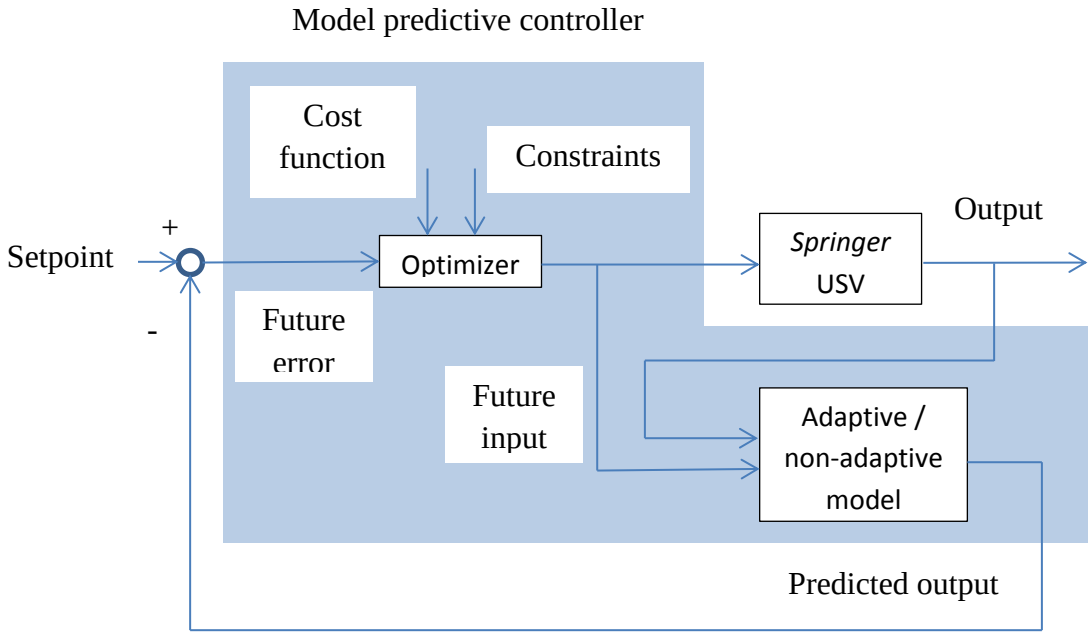
$$J = \sum_{i=1}^{H_p} e(k+i)^T Q\, e(k+i) + \sum_{i=1}^{H_c} \Delta u(k+i)^T R\, \Delta u(k+i) \tag{5}$$

subject to,

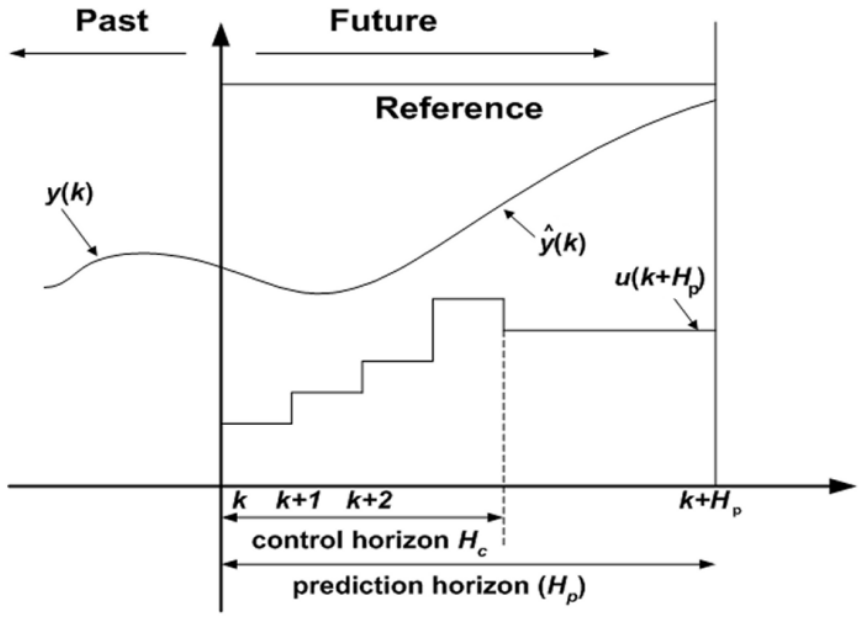
$$\Delta u^l \le \Delta u(k+i) \le \Delta u^u \tag{6}$$

where $e(k) = \hat{y}(k) - r(k)$ is the prediction error, or difference between the predicted vessel output $\hat{y}$ and the reference trajectory r. The superscripts l and u represent the lower and the upper bounds respectively. Q is the weight on the prediction error, and R the weight on the change in the input Δu. H_p is the prediction horizon or output horizon, and H_c the control horizon. The general structure of MPC is shown in Fig 4a.

Minimisation of J with respect to u yields the optimal controller output sequence u_{opt} over the prediction horizon. This ensures that the future error is minimised.



(a) general structure of MPC



(b) general strategy of MPC

Fig 4 MPC (a) general structure (b) general strategy

The general strategy of MPC is illustrated by Fig 4b. As seen in the above Fig 4, at any given instant of time k, the controller looks into the reference set point and exploits the internal model to produce USV output over a period of prediction horizon, into the future.

Corresponding control effort, which yields an optimum value of the cost function, is computed over a period of control horizon. The first control action is applied to the plant.

As time progresses to the next instant of time $(k+1)$, a receding horizon strategy is employed and predicted vessel output and the corresponding control sequences are calculated over the period of prediction and control horizon respectively. In the above Fig 4b, the predicted output and the corresponding optimum input over a horizon H_p are shown, where $u(k)$ is the optimum input, $\hat{y}(k)$ is the predicted output, and $y(k)$ the USV output.

Autopilots based on MPC have a significant advantage over the fixed gain controllers as the controller is designed at every sampling instant. Previous studies by [21], [22] further demonstrate the improved performance of MPC in comparison with standard approaches such as quadratic Gaussian based controllers. The MPC controller also incorporates the actuator limitations of the vessel as optimization constraints. These are given by

$$|n_d| \leq 300\, rpm \text{ and } |\Delta n_d| \leq 20\, rpm \tag{7}$$

that is, a limitation both on the maximum absolute value and on the change of the *rpm* of the motors from one sampling instant to the next. The parameters of the MPC algorithm used are $H_p = 50$ and $H_c = 3$, as these values were necessary to tune the controller, and the weights $Q = 1$ and $R = 0.1$ for the cost function were chosen.

To cope with the sudden change in dynamics, adaptive MPC schemes based on three of the following adaptive algorithms were investigated in this study :

(i) Gradient descent
(ii) Least squares
(iii) Weighted least squares

3.1 Gradient Descent

The gradient descent algorithm tries to minimise a function by following its slope in small steps and provides an updated model of the *Springer* online. It can be visually summarised in Fig 5.

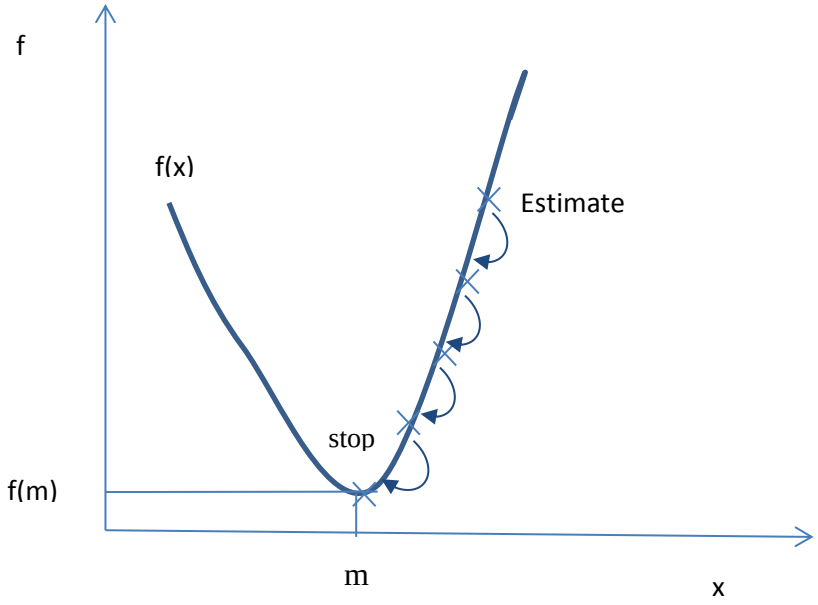


Fig 5 General steps involved in a gradient descent algorithm

The steps involved in a gradient descent algorithm can be summarized as follows:

i. Start with an estimated point
ii. Determine a descent direction
iii. Choose step size
iv. Update (until stopping criteria are reached)

An ARX model of the plant can be represented by the following set of equations:

$$\begin{aligned} &A(z)y_t = B(z)u_t \qquad t \geq 0 \\ &A(z) = 1 + a_1 z + \cdots + a_p z^p, \quad p \geq 0 \\ &B(z) = b_1 z + \cdots + b_q z^q, \quad q \geq 1 \end{aligned} \tag{8}$$

Where A(z) and B(z) are the unknown coefficients of the polynomials and u_t represents the system input. The unknown parameters can be grouped together as follows :

$$\theta = [-a_1 \ldots . -a_p b_1 \ldots b_q]^\tau \tag{9}$$

The parameter θ is estimated by minimising the following criterion

Cost function

$$J_t(\theta) = \frac{1}{2t}\sum_{i=0}^{t}(y_{i+1} - \theta^\tau \phi_i)^2 \tag{10}$$

GD is utilised to obtain the values of θ based on the following equation:

$$\theta_{k+1} = \theta_k - \alpha_{GD}\nabla J(\theta_k) \text{ for k = 0 to number of iterations} \tag{11}$$

α_{GD} is the learning rate and

$$\nabla J(\theta_k) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta(1)} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta(p+q)} \end{bmatrix} \tag{12}$$

The gradient is calculated from the partial derivative of cost function at θ with respect to the corresponding component of θ. The numerical approximation to the partial derivative $\frac{\partial J(\theta)}{\partial \theta(l)}$ is given by the following equation.

$$\frac{\partial J(\theta)}{\partial \theta(l)} = \frac{J(\theta_l^+) - J(\theta_l^-)}{2\delta_{GD}} \tag{13}$$

Where $J(\theta_l^+)$ and $J(\theta_l^-)$ is the cost of θ_l^+ and θ_l^- correspondingly and the values of θ_l^+ and θ_l^- are computed as follows

$$\theta_l^+ = \begin{bmatrix} \theta_{(1)} \\ \vdots \\ \theta_{(l)} + \delta_{GD} \\ \vdots \\ \theta_{(p+q)} \end{bmatrix} \qquad \theta_l^- = \begin{bmatrix} \theta_{(1)} \\ \vdots \\ \theta_{(l)} - \delta_{GD} \\ \vdots \\ \theta_{(p+q)} \end{bmatrix} \tag{14}$$

Calculate

$$\Delta\theta = |\theta_{k+1} - \theta_k| \tag{15}$$

While $\Delta\theta \cong 0$ convergence is reached (or) reached maximum number of iterations, the last best value is taken. This can be visually represented by a flow chart as shown in Fig 6.

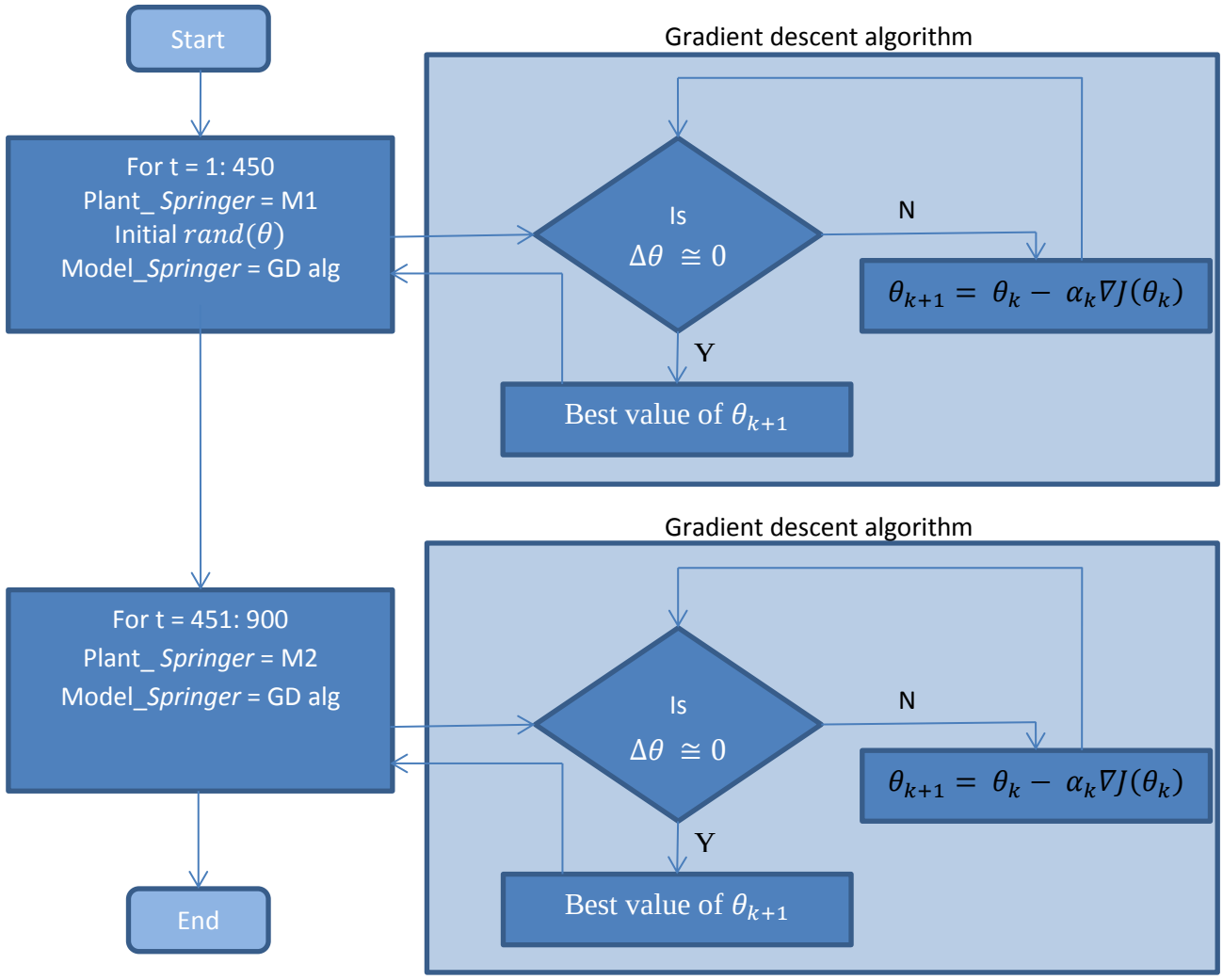


Fig 6 Gradient descent algorithm flow chart for *Springer* USV

A wide range of parameters were tested for gradient descent. The optimum parameters of gradient descent which provided a valid model of the *Springer* are shown in Table 1 as follows :

Table 1 Parameters of Gradient descent for *Springer* USV

Parameters	Values
α_{GD} learning rate (Gradient Descent)	0.000000001
Max no iterations	15000
δ_{GD}	1e-15

Gradient descent provides a simple solution. However, it has certain drawbacks.

1. Starting point: the starting point determines, how long the algorithm takes to converge to a solution and usually this point is chosen arbitrarily.
2. Step size: if large step sizes are chosen, the algorithm might miss the minima point and provide a bad result. On the other hand, if too small step size is chosen, too many

unnecessary steps of computation have to be repeated and the whole process becomes inefficient.

3. Despite all these issues, the major drawback of this method is that the solution can get stuck in local minima at times and there is no guarantee that the solution reached was the global minima.

3.2 Least Squares :

A common and natural way to obtain a model from a set of data is least squares (LS). The model of the plant can be expressed by the following linear regression model

$$y_{t+1} = \theta^{\tau}\phi_t + w_{t+1} \tag{16}$$

Where θ is the unknown parameter vector,
y_t = observation / system output
ϕ_t = regressor
w_t = noise processes / sequences

The parameter θ is estimated by minimising the following criterion

$$J_t(\theta) = \frac{1}{2}\sum_{i=0}^{t}(y_{i+1} - \theta^{\tau}\phi_i)^2 \tag{17}$$

An ARX model of the plant can be represented by the following set of equations:

$$\begin{aligned} &A(z)y_t = B(z)u_t \qquad t \geq 0 \\ &A(z) = 1 + a_1 z + \cdots + a_p z^p, \quad p \geq 0 \\ &B(z) = b_1 z + \cdots + b_q z^q, \quad q \geq 1 \end{aligned} \tag{18}$$

Where A(z) and B(z) are the unknown coefficients of the polynomials and u_t represents the system input. The unknown parameters can be grouped together as follows :

$$\theta = [-a_1 \ldots . -a_p b_1 \ldots b_q]^{\tau} \tag{19}$$

The recursive LS algorithm is as follows

$$\theta_{t+1} = \theta_t + L_t(y_{t+1} - \theta_t^{\tau}\phi_t) \tag{20}$$

$$L_t = \frac{P_t\phi_t}{1+\phi_t^{\tau}P_t\phi_t} \tag{21}$$

$$P_{t+1} = P_t - \frac{P_t\phi_t\phi_t^{\tau}P_t}{1+\phi_t^{\tau}P_t\phi_t} \tag{22}$$

$$\phi_t = [y_t \ldots . y_{t-p+1}\, u_t \ldots u_{t-p+1}\ ]^{\tau} \tag{23}$$

The standard parameter of least squares α_{LS} is one and it has been utilised with *Springer* USV.

Some of the issues regarding the standard LS are as follows:

1. The estimates may not converge (or even may not be bounded). i.e., LS does not have self-convergence property.
2. The estimated models may not be uniformly controllable.

3.3 Weighted least squares :

One of the key advantages of weighted least squares (WLS) is that it has a self-convergence property [23]. Irrespective of the control law design, this algorithm converges to a particular arbitrary vector. The 'universal convergence' result eliminates the analysis of stochastic adaptive control systems. This is achieved by slowly decreasing the weights of the system. Then the model hence obtained is uniformly controllable and enables to create a general framework for an adaptive robust control system design for *Springer*. In this method, the parameter θ is estimated by minimising the following criterion

$$J_t(\theta) = \frac{1}{2}\sum_{i=0}^{t} \propto_i (y_{i+1} - \theta^{\tau}\phi_i)^2 \tag{24}$$

where $\propto_i \geq 0$ is a weighting sequence. It enables allocation of different weights to different measurements. In a closed loop, the values of actual observation (y_t), $\mathrm{regressor}(\phi_t)$ are unknown. Decreasing $\propto_i$ decreases the effect of instability and lack of excitation. Moreover, decreasing the rate of $\propto_i$ ensures that WLS enjoys similar good asymptotic properties as the standard LS. If the ARX model of the plant are represented by the equations (3) and (4).

The recursive WLS algorithm is as follows

$$\theta_{t+1} = \theta_t + L_t(y_{t+1} - \theta_t^{\tau}\phi_t) \tag{25}$$

$$L_t = \frac{P_t\phi_t}{\alpha_t^{-1} + \phi_t^{\tau}P_t\phi_t} \tag{26}$$

$$P_{t+1} = P_t - \frac{P_t\phi_t\phi_t^{\tau}P_t}{\alpha_t^{-1} + \phi_t^{\tau}P_t\phi_t} \tag{27}$$

$$\phi_t = [y_t \ldots . y_{t-p+1}\ u_t \ldots u_{t-p+1}\ ]^{\tau} \tag{28}$$

$$\alpha_t = \frac{1}{f(r_t)} \tag{29}$$

$$r_t = \|P_o^{-1}\| + \sum_{i=0}^{t}\|\phi_i\|^2 \tag{30}$$

The optimum parameters of WLS which provided a valid model of the *Springer* are in Table 2, as follows:

Table 2 Parameters of WLS for *Springer* USV

Parameters	Values
α_{WLS}	0.5
δ_{WLS}	1

4 Results and discussion

As stated earlier, *Springer* offers a standard displacement of 0.6 tonnes. Under the present circumstances, the dynamic model of the *Springer* vehicle is given by the equations (7) and (8). The objective of this study has been to develop a robust adaptive autopilot for the *Springer* USV which will be capable of handling changes in the mass of the USV.

4.1 Step response

Before attempting to use the models obtained from the above methods in closed loop with a model predictive controller, it is imperative to ensure its integrity. Hence, the step responses were obtained as follows:

The step response of the original plant in closed loop as described in equations (3) and (4) is shown in the following Fig 7 and the corresponding step response characteristics are summarised in Table 3.

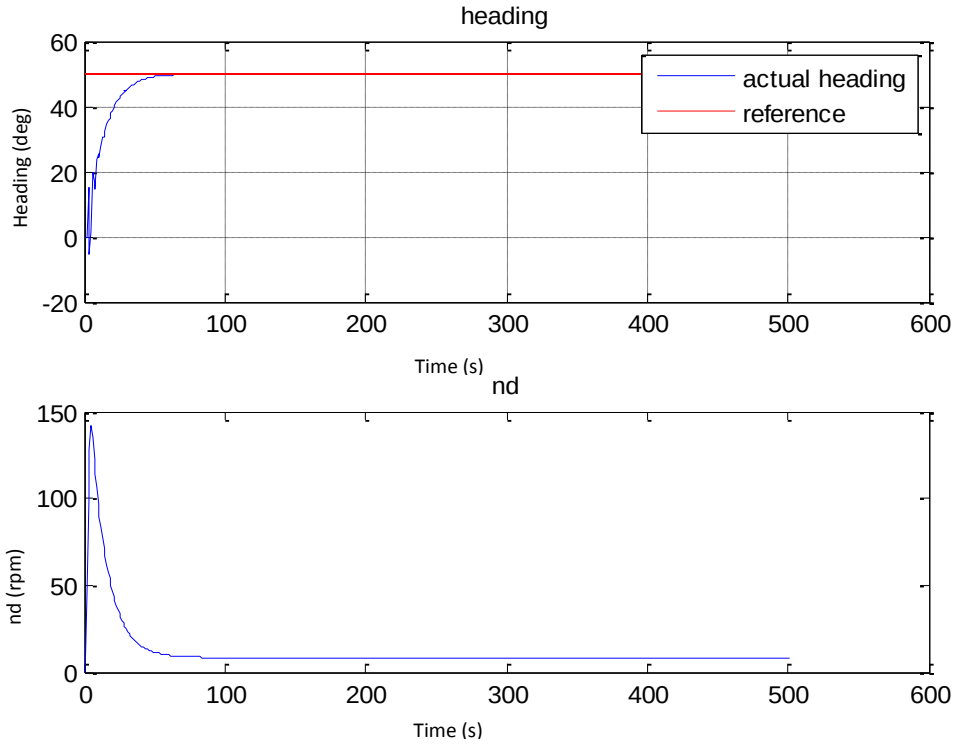


Fig 7 Step response with original parameters of the plant as described in equation (4)

Table 3 Step response of original *Springer* USV

Parameters	**Values**
Rise Time (s)	27.35
Settling Time (s)	47.62
Settling Min (s)	45.13
Settling Max ($)^0$	50.00
Overshoot ($)^0$	5.99e-13
Undershoot (%)	10.63
Peak ($)^0$	50.00
Peak Time (s)	481

As discussed in the earlier sections, in the event of change in mass of the system, usually it ranges from 0% to 20% of the total mass of the USV. However, during extreme weather conditions or sudden change in mass of the USV as in search and rescue operation (or payload deployment from USV) the worst case scenario could offset the system parameters considerably. Whilst it is imperative to push the limits of robustness of the autopilot, it will become a fallacy to endeavour a solution for practically impossible situations (such as 100% change in mass for example). Hence, to test the endurance of the robust adaptive autopilot, different changes were studied. Initial trials conducted at Roadford reservoir, Devon, UK indicate that the following system matrix A represented by equation (31) represents a case for 50% change in mass of the *Springer*. This system matrix was chosen to highlight the effectiveness of different methods to cope with such a change (should such a severe change occur in reality). When such a change in dynamics of the plant occurs, corresponding significant changes in the step response were also observed as illustrated in Fig 8 and the corresponding step response characteristics are summarised in Table 4.

$$A = \begin{bmatrix} 0.0186 & 1 & 0 & 0 \\ 0.0501 & 0 & 1 & 0 \\ 0.1891 & 0 & 0 & 1 \\ 0.4710 & 0 & 0 & 0 \end{bmatrix} \tag{31}$$

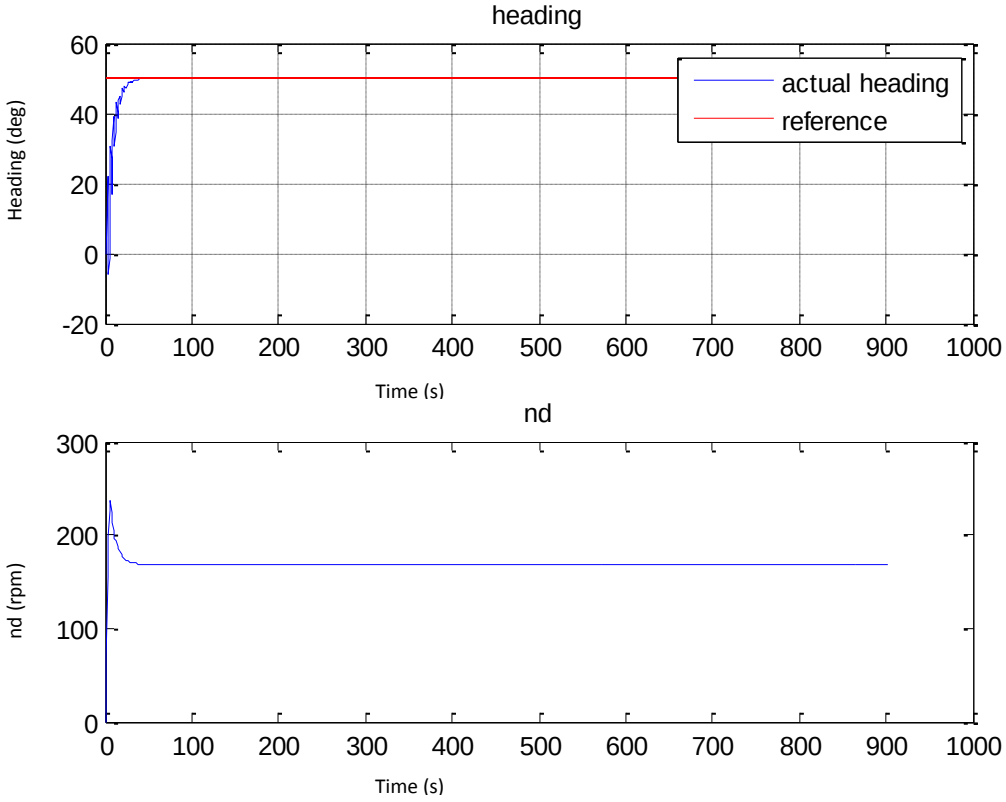


Fig 8 Step response for the new system dynamics of the plant

Table 4 Step response of *Springer* USV after the change in system dynamics

Parameters	**Values**
Rise Time (s)	14.75
Settling Time (s)	29.51
Settling Min (s)	42.75
Settling Max ( ) ^{0}	50.00
Overshoot ( ) ^{0}	2.66e-13
Undershoot (%)	11.94
Peak ( ) ^{0}	50.00
Peak Time (s)	335

The closed loop performance of the controller and the USV was analysed against different reference signals and this will serve as a bench mark to compare the other adaptive algorithms discussed above. The results are shown in the following Fig 9.

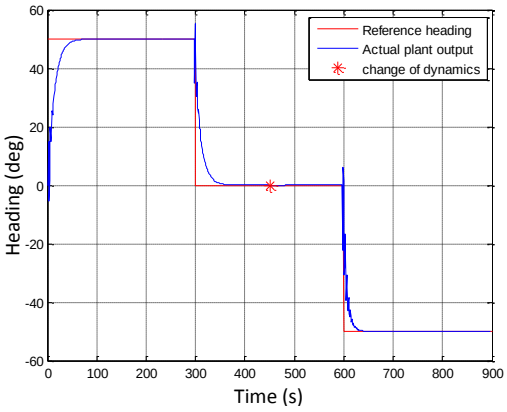


Fig 9a change of system dynamics at 451 s

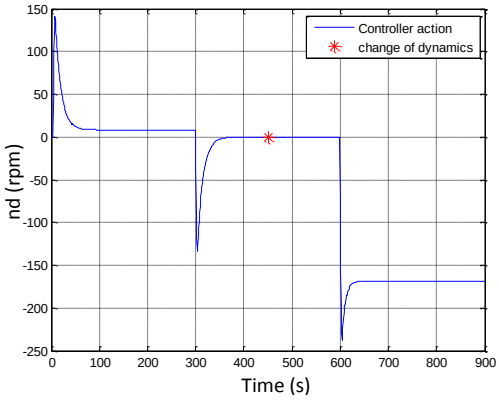


Fig 9b Controller action for change of system dynamics at 451 s

Fig 9 Change of system dynamics (a) plant output and three reference signals (b) controller action

From the above Fig 9 it can be observed that the change of mass did not make any visible difference as the reference was already at zero during the change in dynamics. So a different reference was chosen and changes were obvious as presented in the following Fig 10

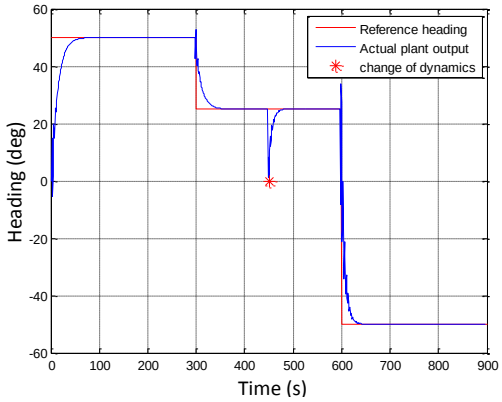


Fig 10a change of system dynamics

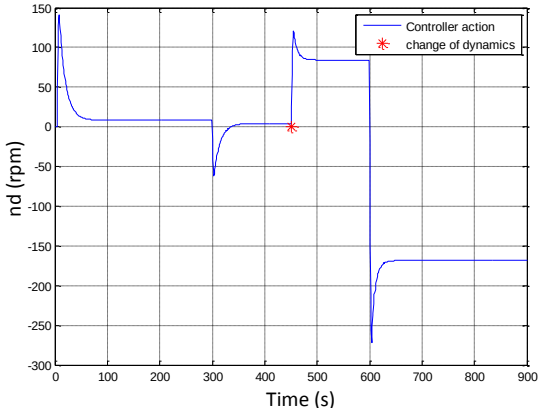


Fig 10b controller action for change of system dynamics

.

Fig 10 Change of reference headings (a) plant output (b) controller action

Once the controller performance and a frame work to compare the further results were established, the online models obtained from the *Springer* USV were utilised by the MPC controller and the results are presented in this section.

4.2 Gradient descent and MPC

Gradient descent algorithm as described in the previous section 3.1 was utilised in conjunction with the MPC as illustrated in Fig 4(a). The initial θ values were randomly initialised during initiation of the algorithm. As the initial values were randomly chosen, several runs were carefully examined and the following two cases are presented here to illustrate the range of outputs obtained from this course of action.

The following Figs extol the impact of initial values of θ. As explained the flowchart (Fig 6), the algorithm obtains a new model of the USV at every sampling instant. In effect, it runs 15,000 iterations for every sampling instant and hence to complete one set of algorithm; it took 1 hour and 18 minutes approximately.

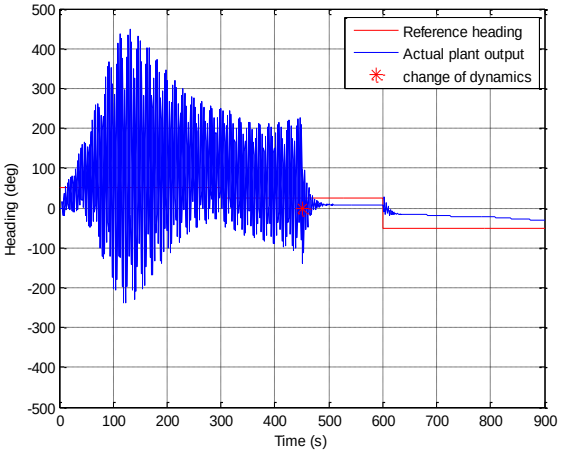


Fig 11a case 1; Plant output

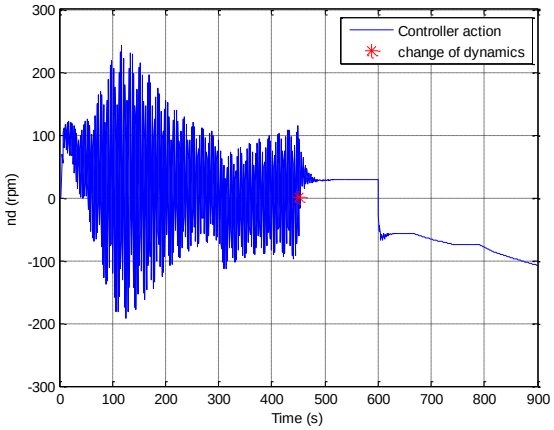


Fig 11b case 1; Controller action

Fig 11 case 1; Gradient Descent and MPC controller (a) plant output (b) controller action

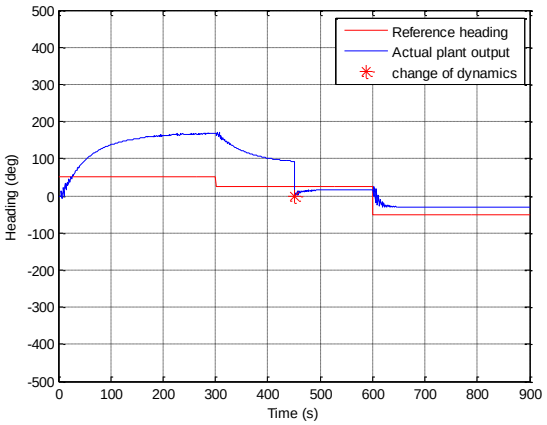


Fig 12a case 2; Plant output

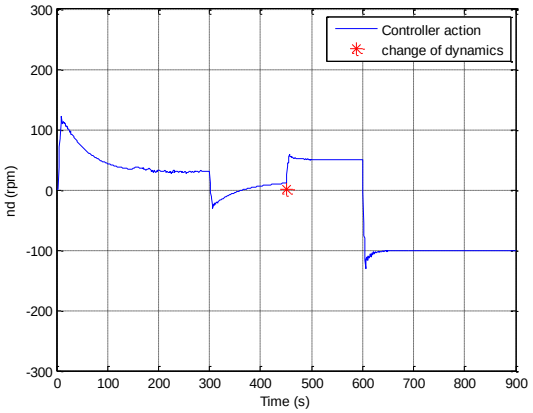


Fig 12b case 2; Controller action

Fig 12 case 2; Gradient Descent and MPC controller (a) plant output (b) controller action

Despite being computationally expensive, the results from the gradient descent were not that promising. Due to the true random nature of the initial values chosen, no two runs were the same. The impact of the initial random values can be clearly illustrated in the 2 above cases represented in Fig 11 and 12. This implied that the initial θ value assignment had a significant effect on subsequent behaviour. Hence the standard gradient descent was modified slightly as follows. Instead of total random assignments of the initial θ values, 25% of the true values of the original plant were chosen as initial θ. This implies that the system will still cope with 75% error in the initial θ values. Moreover, instead of computing the gradient algorithm for every sampling instant, it was initiated only once for every 100 seconds. The changes in the results were dramatic as can be seen in the following Fig 13.

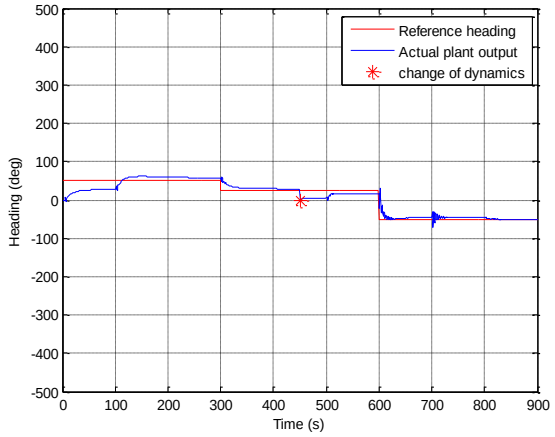


Fig 13a case 3; Plant output

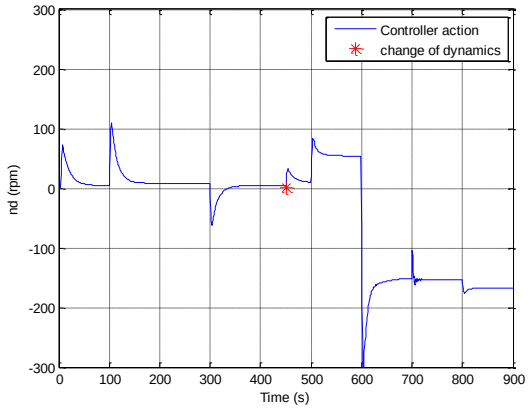


Fig 13b case 3; Controller action

Fig 13 case 3; Modified Gradient Descent and MPC controller (a) plant output (b) controller action

The improvements in the performance of the autopilots are visually evident from the Fig 11 – 13. However, to measure the improvements numerically, the average controller energy(ACE) and mean square error (MSE) were utilised. In a discrete form, these two parameters can be calculated by the following equations

$$ACE = \frac{1}{M}\sum_{t=1}^{M}[u(t)]^2 \tag{32}$$

$$MSE = \frac{1}{M}\sum_{t=1}^{M}[y(t) - r(t)]^2 \tag{33}$$

where u(t) is the controller effort at an instant of time t, M is the total number of samples, y(t) is the output from the USV in degrees and r(t) is the reference angle which the USV is supposed to track in this study.

The corresponding values of ACE and MSE were calculated for the different options illustrated by Fig 11-13 and the results are summarised in the Table 5. In pursuit of further improvements, the next section details the results obtained by utilising LS with MPC.

4.3 Least squares and MPC

Given, its simplicity and lean computation, least square is seen as a natural choice to obtain model of the vessel for a given set of input, output data. Initial performance was satisfactory and was able to work well in conjunction with MPC controller. However, as soon as there was a change in the dynamics of the plant, then it was no longer able to track the reference in a satisfactory manner. This can be clearly observed from the following Fig 14.

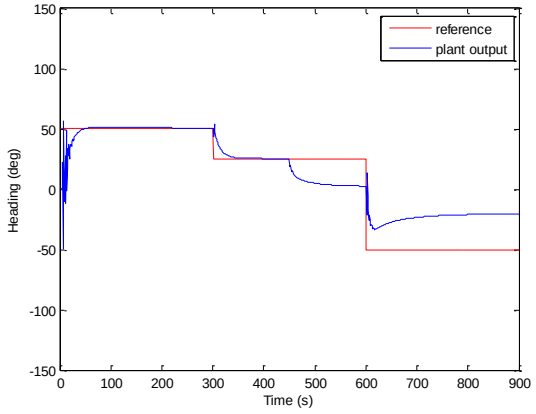


Fig 14a plant output

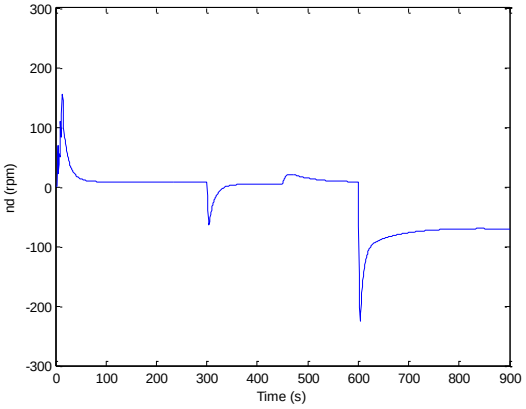


Fig 14b controller action

Fig 14 Least squares and MPC controller (a) plant output (b) controller action

The corresponding values of ACE and MSE were calculated and the results are summarised in the Table 5.

4.4 Weighted least squares and MPC

Weighted least squares certainly has theoretical advantages over the least squares, as mentioned in previous section 3.3. In reality, there was only a marginal improvement of the results and hitherto, the change in the dynamics had offset the reference tracking ability of the autopilot. This can be seen from the following Fig 15.

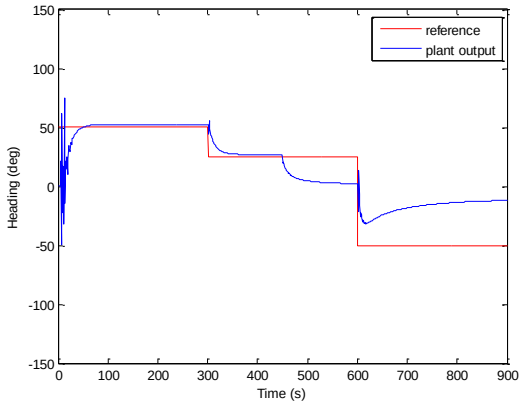


Fig 15a plant output

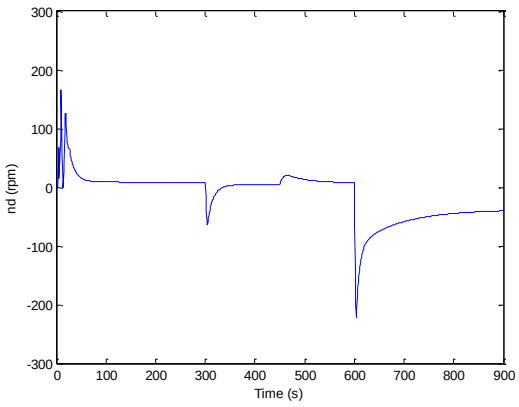


Fig 15b controller action

Fig 15 Weighted least squares and MPC controller (a) plant output (b) controller action

To tackle this problem, the WLS algorithm was reinitiated when there was a change in the dynamics of the system. In the real world it may not be possible to have *a priori* knowledge of the change in the system dynamics. Hence decision making logic was implemented to detect the change in the dynamics and reinitialise the WLS algorithm, only when these conditions were satisfied.

$$\Delta\theta_t = \theta_t - \theta_{t-1} > 80 \text{ to } 120\ \% \tag{34}$$

From the values of the individual components of θ, it was observed that a change of 80 to 120% or more signified a sudden change in system dynamics. At the same time, the system was still being initialised with random θ values and huge changes of $\Delta\theta$ were common during the initialisation. If this was wrongly detected as change in system dynamics, the algorithm will be reset continuously and yield very poor results. Hence, to ensure that the algorithm was only reinitialised when there was a change in system dynamics, the above criteria of $\Delta\theta$ was used in conjunction with the following two additional criterions

$$\Delta\theta_{t-1} = \theta_{t-1} - \theta_{t-2} \cong 0 \; or < 10\,\% \tag{35}$$

$$\Delta\theta_{t-2} = \theta_{t-2} - \theta_{t-3} \cong 0 \; or < 10\,\% \tag{36}$$

The values of $\Delta\theta_{t-1}$ and $\Delta\theta_{t-2}$ was usually detected to be approximately zero or it reached a maximum value of 10% in some cases. Once the above three conditions were satisfied, it was deemed appropriate to reinitialise the WLS algorithm.

This allows time for the WLS algorithm to reach a steady state with the random initial conditions and reinitiates the algorithm when there is a change in the dynamics. This is indicated by the following Fig 16 where the norm(P) indicates that it has been reset only once after initialisation.

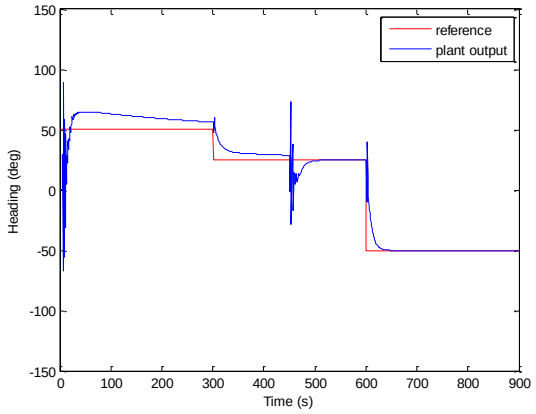


Fig 16a WLS re-initialised Plant output

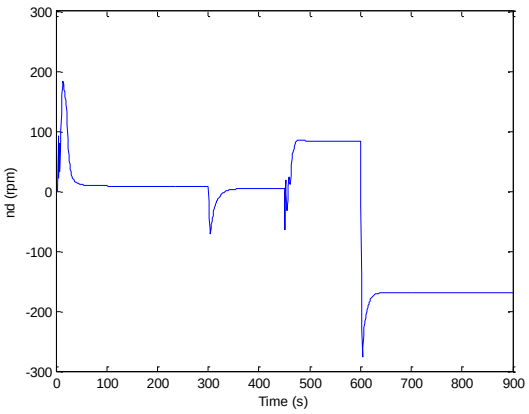


Fig 16b WLS re-initialised controller action

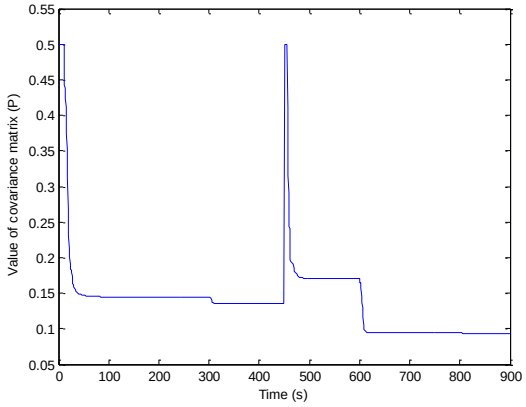


Fig 16c Re-initialised covariance matrix (P)

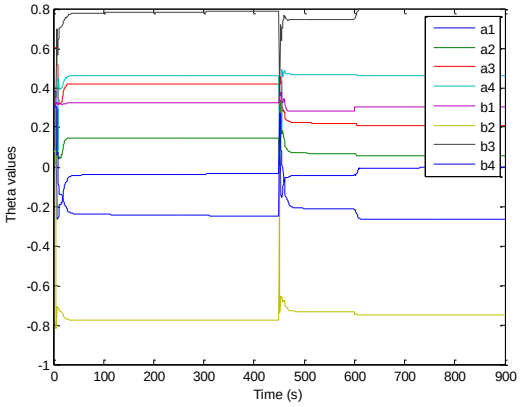


Fig 16d θ values

Fig 16 Weighted least squares and MPC controller (a) plant output (b) controller action (c) re-initialised covariance matrix (P) (d) θ values

Once improved results were obtained by reinitialising the algorithm, it was contemplated, if reinitialising the algorithm periodically might improve the performance further. These results can be seen in the following Figs, where the system was reset periodically for every 50 s. Re-initialisation caused the θ values and the covariance matrix P (equation (25)) to be reset every time. Hence spikes at change of dynamics reached values more than 100. So this approach was deemed unfit for real-time application. Nevertheless, it paved the way to obtain the subsequent steps.

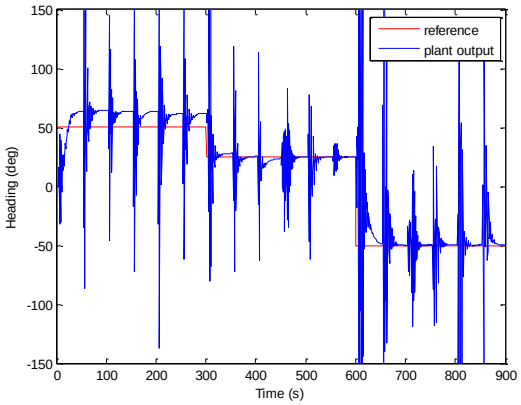


Fig 17a WLS re-initialised Plant output

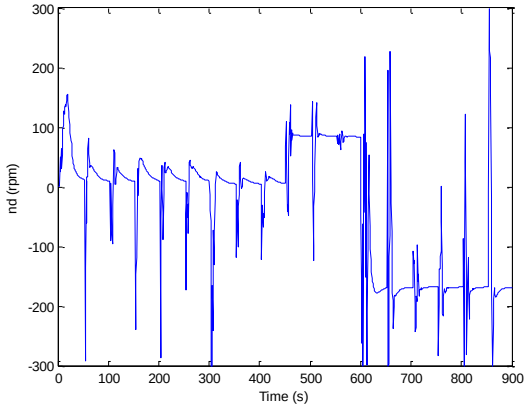


Fig 17b WLS re-initialised controller action

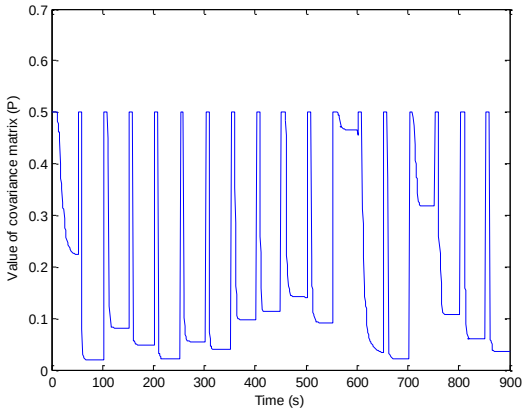


Fig 17c Re-initialised covariance matrix (P)

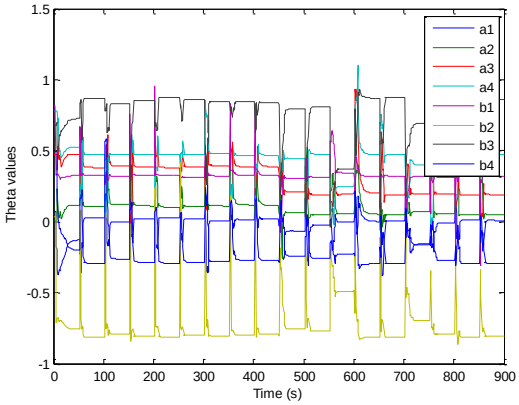


Fig 17d θ values

Fig 17 WLS reinitialised for every 50 sec (a) plant output (b) controller action (c) re-initialised covariance matrix (P) (d) θ values

Instead of re-initialising the entire algorithm, much better performance was obtained by resetting the values of the covariance matrix P, at periodic intervals and by keeping the θ values continuous. The results are obvious from the following Fig 18. Additionally this approach eliminates the need for having a decision making process (which inherently carries the risk of resetting the system inadvertently). Thus by the above said modifications, the robust adaptive autopilot capable of handling changes in the system dynamics has been realised.

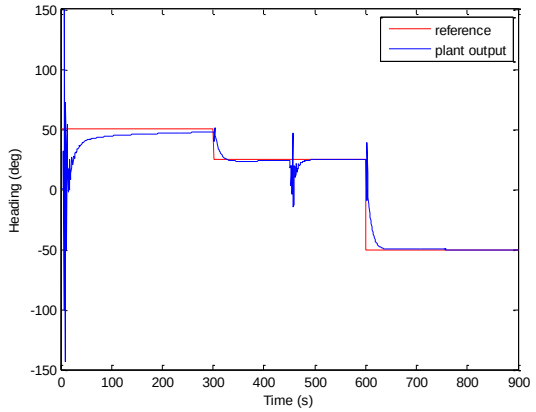


Fig 18a (P) re-initialised Plant output

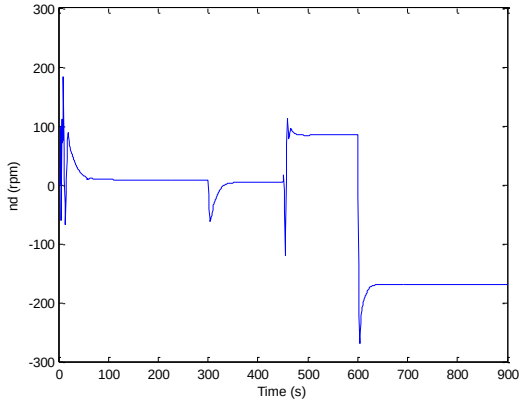


Fig 18b controller action

Fig 18 Covariance matrix (P) reinitialised for every 50 sec, with continuous θ (a) plant output (b) controller action

Once this approach yielded satisfactory results, it was time to retune the periodic intervals at which the covariance matrix (P) was reset. When the window length was increased to 100 seconds the performance deteriorated as can be seen from the following Fig 19.

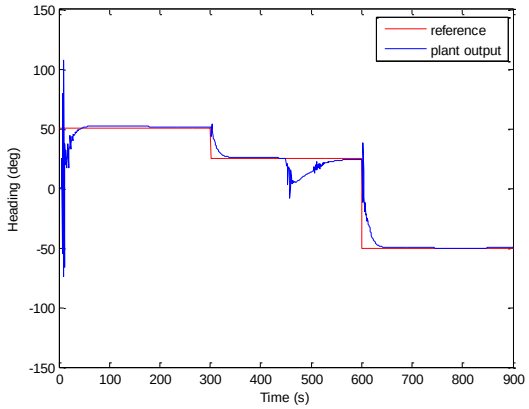


Fig 19a (P) re-initialised Plant output

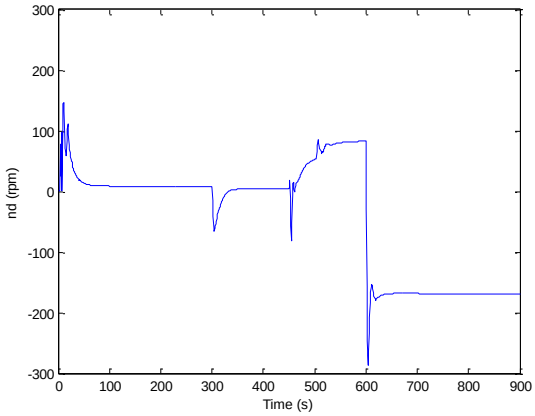


Fig 19b controller action

Fig 19 Covariance matrix (P) reinitialised for every 100 sec, with continuous θ (a) plant output (b) controller action

Then the window of time was reduced to different time periods and the performance deteriorated below 25 seconds. Hence the optimum time to reset the covariance matrix for *Springer* USV was found to be 25 seconds. The results can be found as follows in Fig 20.

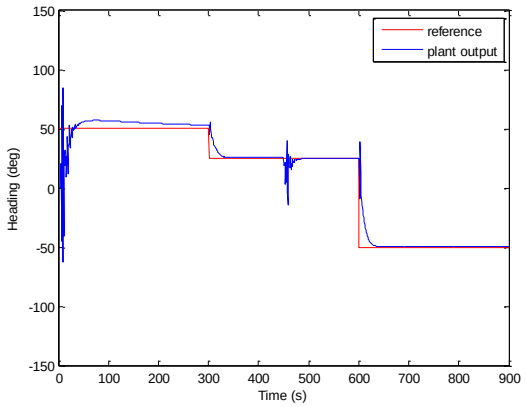


Fig 20a (P) re-initialised Plant output

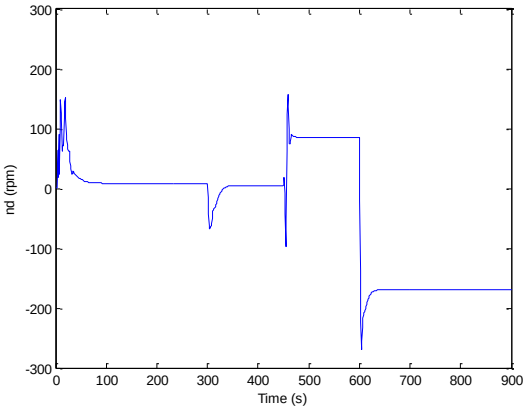


Fig 20b controller action

Fig 20 Covariance matrix (P) reinitialised for every 25 sec, with continuous θ (a) plant output (b) controller action

These results from the modified WLS were benchmarked against the real values of θ and the results were strikingly very similar. Moreover, the WLS handled the change in the dynamics better than providing the real values to the controller directly.

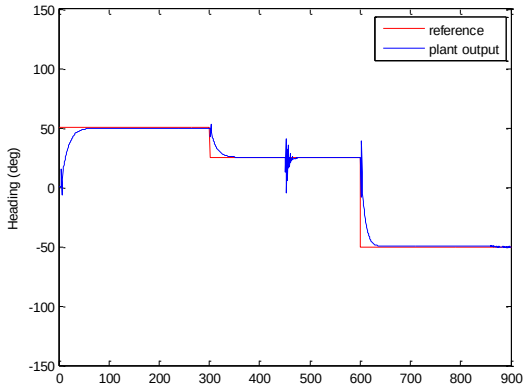


Fig 21a plant output

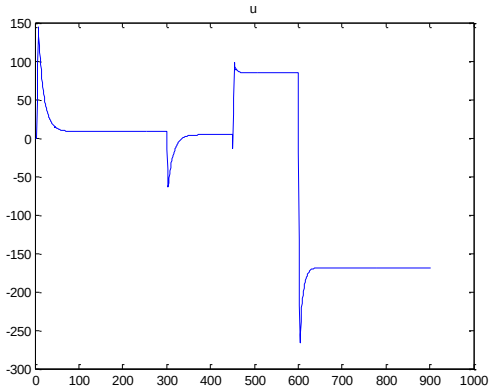


Fig 21b controller action

Fig 21 Weighted least squares with real θ and MPC controller (a) plant output (b) controller action

The corresponding values of ACE and MSE were calculated for the different options illustrated by Fig 15-21 and the results are summarised in the Table 5 as follows.

Table 5 Comparison of performance of autopilots (for *Springer* USV)

Main algorithm used	**Different cases**	**ACE (rps^2)**	**MSE (deg^2)**
Gradient descent	case 1; Gradient descent	1.8273	1.1968e+04
	case 2; Gradient descent	1.3253	4.6621e+03
	case 3; Modified gradient descent	2.6839	189.0032
Least squares	Standard Least squares	0.6253	408.6256
Weighted least squares	Standard weighted least squares	0.4827	517.1449
	Reinitialised during a change in dynamics	3.1483	148.2021
	Reinitialised for every 50 sec	3.4617	1.8671e+03
	Covariance matrix (P) reinitialised for every 50 sec, with continuous θ	3.1026	179.6000
	Covariance matrix (P) reinitialised for every 100 sec, with continuous θ	2.9731	125.5104
	Covariance matrix (P) reinitialised for every 25 sec, with continuous θ	3.1332	107.5875
	WLS with real θ	3.1126	68.9770

From the above table, it can be observed that WLS with no modifications clearly has a large tracking error. This is considerably reduced by reinitiating the algorithm during a change in dynamics of the USV. Further attempts were made to improve the performance by reinitialising the algorithm periodically. On the outset, it seemed counterproductive as the error values increased significantly. On closer analysis, it became clear that such behaviour was due to frequent random initialisation and the transient response characteristics. Instead of solving a problem, now it served as additional burden on the system. Significant improvements were achieved by keeping the θ values continuous and reinitialising only the covariance matrix. Once this behaviour was understood, then marginal gains were made by varying the size of the time frame window to carry out the initialisation. The values from these cases are also summarised in the above table 5. A time frame of 25 seconds seems to be the optimum time frame to the reinitialise the parameters discussed above.

This novel approach enables the autopilot to cope well with significant changes (approximately 50%) in the system dynamics. Under normal circumstances, the changes in the system dynamics are likely to be 0 to 20%. Hence it is deemed appropriate for *Springer* and is highly recommended to design a robust adaptive autopilot for other such USVs by utilising this innovative approach.

5 Conclusions

The sudden change in the dynamics of a plant causes considerable deterioration (in case of non-adaptive techniques) in the controller performance and remains a major obstacle in accomplishing the desired missions. This problem has been suitably dealt in this study by designing a robust adaptive autopilot for the *Springer* USV. Initially the basic structure of the controller and the performance of the original offline plant dynamics were established. Subsequently, the performance was compared with three suitable methods namely: gradient descent, LS and WLS. Gradient descent is able to provide a solution with very bad tracking performance. Additionally, there is a risk that it might get stuck in the local minima. Hence LS was investigated further. The system stabilises even after the change in dynamics. However, the self-convergence is not guaranteed by the LS approach and the autopilot was not able to cope with change in dynamics. Hence WLS was investigated further. However, applying WLS to solve this problem posed several, severe, initial challenges. To overcome these issues a new approach has been presented here. Random initialisations of θ values and periodically reinitialise the covariance matrix P at intervals of 25 seconds, whilst keeping the intermittent θ values continuous offers a new approach to deal with change in dynamics. It is also worth mentioning that in a standard WLS the recent values are given more weightage and it will make the system unstable if the changes in the immediate values are large. Stability and controllability are guaranteed by giving less weightage to the immediate values and giving more weightage to the past values. Subsequent to these suitable modifications it was found to be the most appropriate to be incorporated in the design of a robust adaptive autopilot. Moreover, this approach also eliminates the need for implementing a decision making algorithm / logic to detect the change in dynamics. Hence by implementing such a technique further problems such as false trigger or no detection of changes is comprehensively circumvented successfully. This approach checks for changes periodically and at the same time manages to be computationally efficient by keeping the continuous θ values. This inventive approach will enable the *Springer* USV and other USVs to cope with change in dynamics and still accomplish the desired missions effectively.

References

[1] Eshel T.: US Navy tests Rafael Spike missiles on unmanned vessels. 31 October (2012) (http:// defense-update.com, accessed 30 May 2013)

[2] Park S., Kim J., Lee W. and Jang C.: A study on the fuzzy controller for an unmanned surface vessel designed for sea probes. *Proc of International Conference on Control, Automation and Systems,* Kintex, Korea, June, 1-4 (2005)

[3] Alves J., Oliveira P., Oliveira R., Pascoal A., Rufino M., Sebastiao L. and Silvestre C.: Vehicle and mission control of the *Delfim* autonomous surface craft. *Proc of 14th Mediterranean Conference on Control Automation,* Ancona, Italy, June, 1-6 (2006)

[4] Elkaim G.H. and Kelbley R.: Measurement based H infinity controller synthesis for an autonomous surface vehicle. *Proc of 19th International Technical Meeting of the Satellite Division of the Institute of Navigation,* Fort Worth, USA, September (2006)

[5] Naeem W., Sutton R. and Chudley J.: Soft computing design of a linear quadratic Gaussian controller for an unmanned surface vehicle. *Proc of 14th Mediterranean Conference on Control Automation,* Ancona, Italy, June, 1-6 (2006)

[6] Ashrafiuon H., Muske K.R., McNinch L.C. and Soltan R.A.: Sliding-mode tracking control of surface vessels. *IEEE Trans on Industrial Electronics*, **55**(11), November, 4004-4012 (2008)

[7] Qiaomei S., Guang R., Jin Y. and Xiaowei Q.: Autopilot design for unmanned surface vehicle tracking control. *Proc of 3rd International Conference on Measuring Technology and Mechatronics Automation*, **1**, January, Shangshai, China, 610-613 (2011)

[8] Sharma SK, Naeem W and Sutton R.: An autopilot based on a local control network design for an unmanned surface vehicle. *Journal of Navigation*, **65**(2), 281-301 (2012)

[9] Naeem, W., Xu, T., Sutton, R. and Tiano, A.: The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring. *Proc Instn Mech Engrs Part M: Journal of Engineering for the Maritime Environment*, **222**(M2), 67-80 (2008)

[10] Annamalai, ASK.: A review of model predictive control and closed loop system identification for design of an autopilot for uninhabited surface vehicles (2012) *Springer Technical Report:* MIDAS.SMSE.2012.TR.005, 2012

[11] Li, Z., and Sun, J.: Disturbance compensating model predictive control with application to ship heading control. *IEEE Trans on Control Systems Technology*, **20**(1), 257-265 (2012)

[12] Liu, J., Allen, R., and Yi, H.: Ship motion stabilizing control using a combination of model predictive control and an adaptive input disturbance predictor. *Proc IMechE Part I: Journal of Systems and Control Engineering*, **225**(5), 591-602 (2011)

[13] Oh, S.R., and Sun, J.: Path following of under actuated marine surface vessels using line-of-sight based model predictive control. Ocean Engineering, **37**(2-3), pp.289-295 (2010)

[14] Naeem, W., Sutton, R., Chudley J., Dalgleish FR., and Tetlow, S.: An online genetic algorithm based model predictive control autopilot design with experimental verification. *International Journal of Control*, **78**(14/20), 1076-1090 (2005)

[15] Perez, T.: Ship motion: Course keeping and roll stabilisation using rudder and fins. *Springer-Verlag*, London (2005)

[16] Maciejowski, JM.: Predictive control with constraints. *Prentice Hall Inc*., London (2002)

[17] Allgower, F., Glielmo, L., Guardiola, C., and Kolmanovsky, I.: Automotive model predictive control. *Springer-Verlag*, Berlin (2010)

[18] Rawlings, JB., and Mayne, DQ.: Model predictive control: Theory and design. *Nob Hill Publishing*, Madison (2009)

[19] Wang, L.: Model predictive control system design and implementation using MATLAB. *Springer-Verlag*, Berlin (2009)

[20] Ljung L.: System Identification, Theory for the User. Second Edition, Prentice Hall, New Jersey, USA (1999)

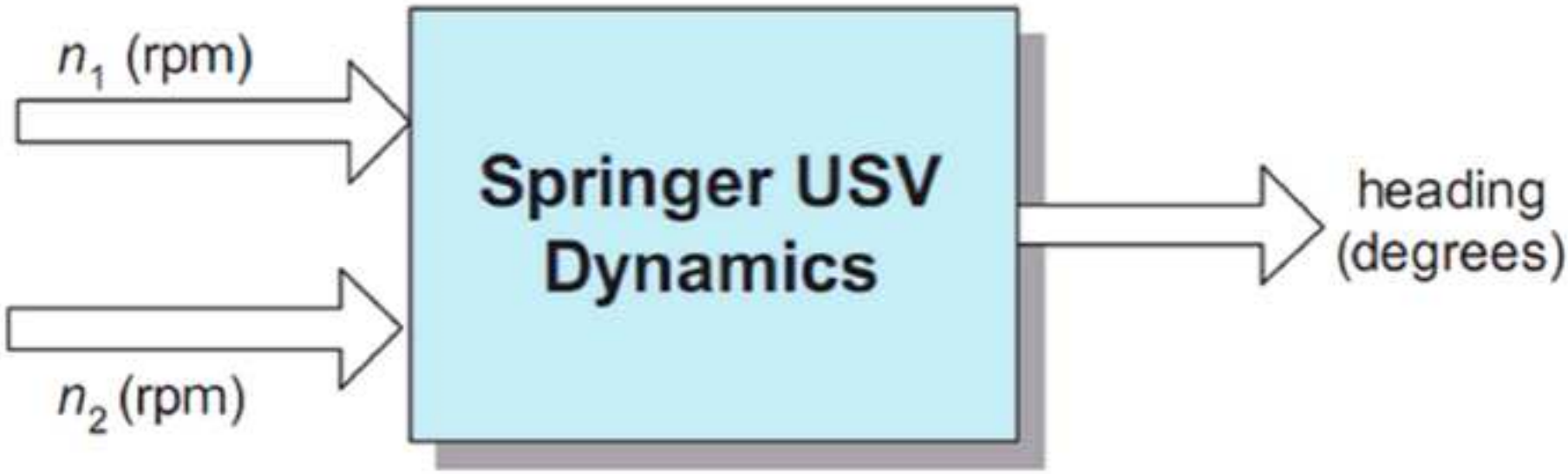
[21] Annamalai, ASK., and Motwani, A.: A comparison between LQG and MPC autopilots for inclusion in a navigation, guidance and control system. *Springer Technical Report*, MIDAS SMSE.2013.TR.006. Plymouth University, Plymouth, UK (2013)

[22] Annamalai, ASK, Motwani, A, Sutton, R, Yang, C, Sharma, SK, Culverhouse P.: Integrated navigation and control system for an uninhabited surface vehicle based on interval Kalman filtering and model predictive control. *Proc of the 1st IET Control and Automation Conference, Conference Aston Lakeside Centre,* Birmingham, UK, June, 4-5 (2013)

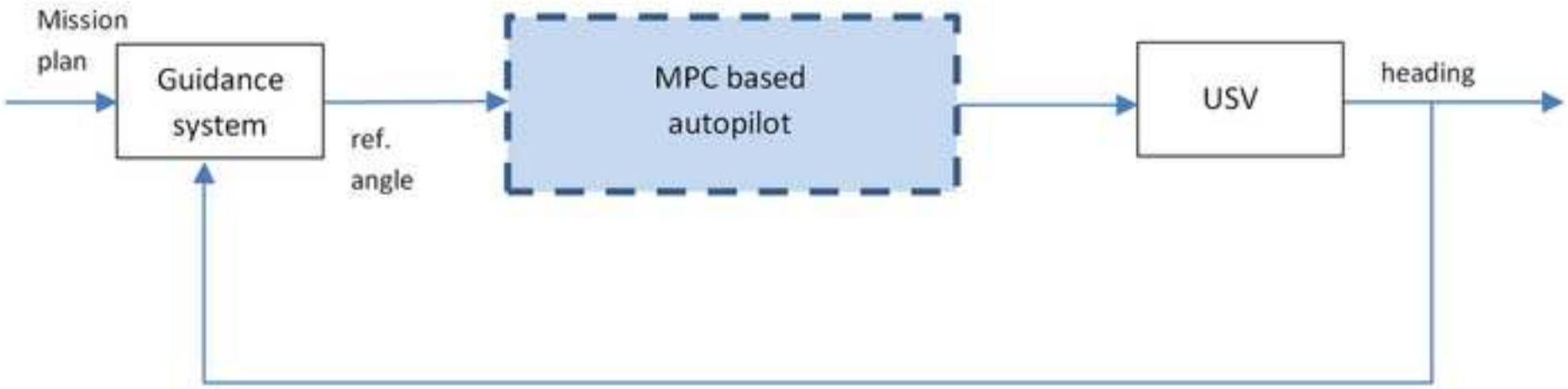
[23] Guo, L.: Self-convergence of weighted least squares with applications to stochastic adaptive control, IEEE transactions on automatic control, **41**(1), January, 79-89 (1996)

Springer USV
Click here to download high resolution image

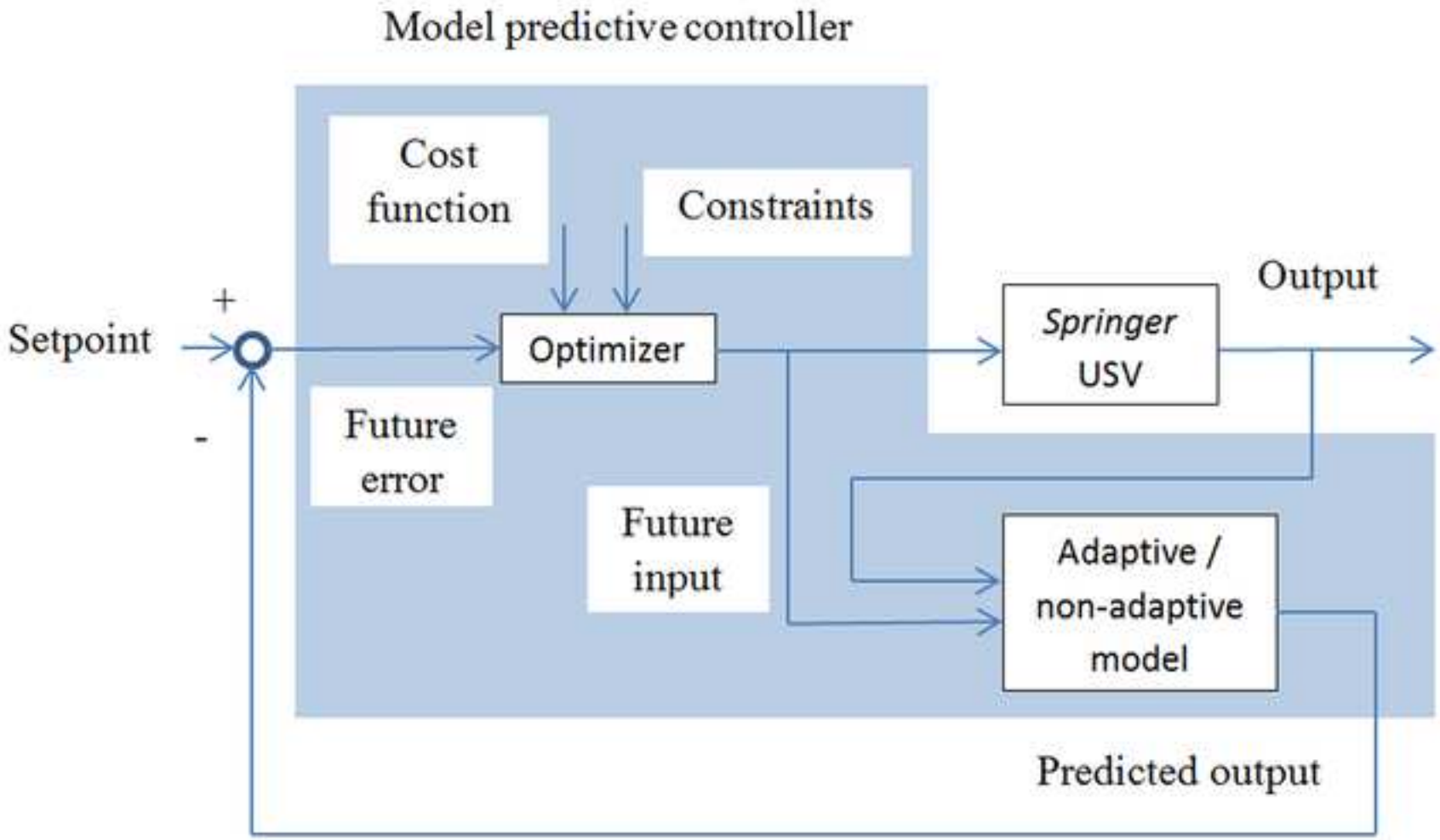
Block diagram representation of Springer USV
Click here to download high resolution image



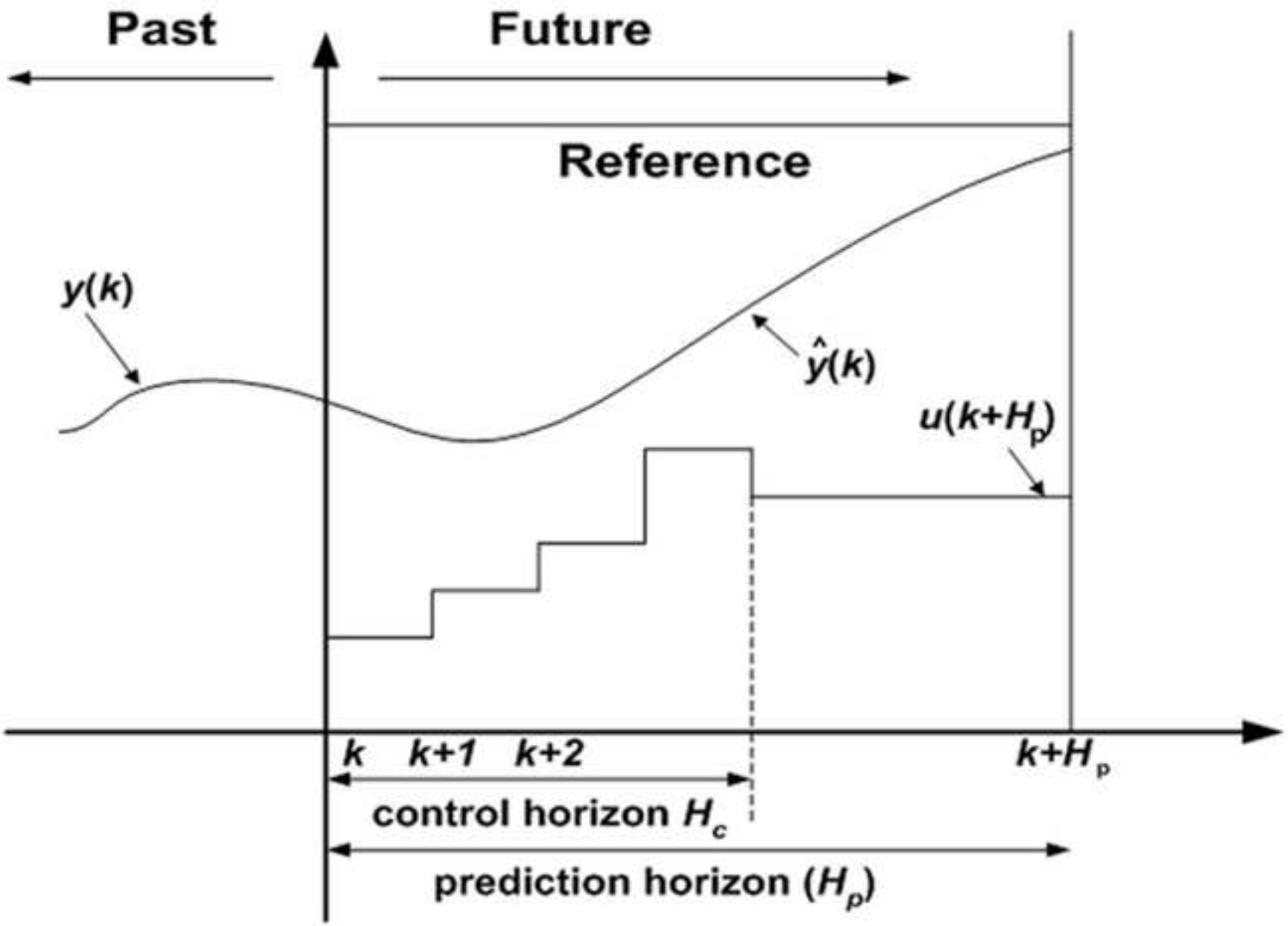
Autopilot of Springer USV
Click here to download high resolution image



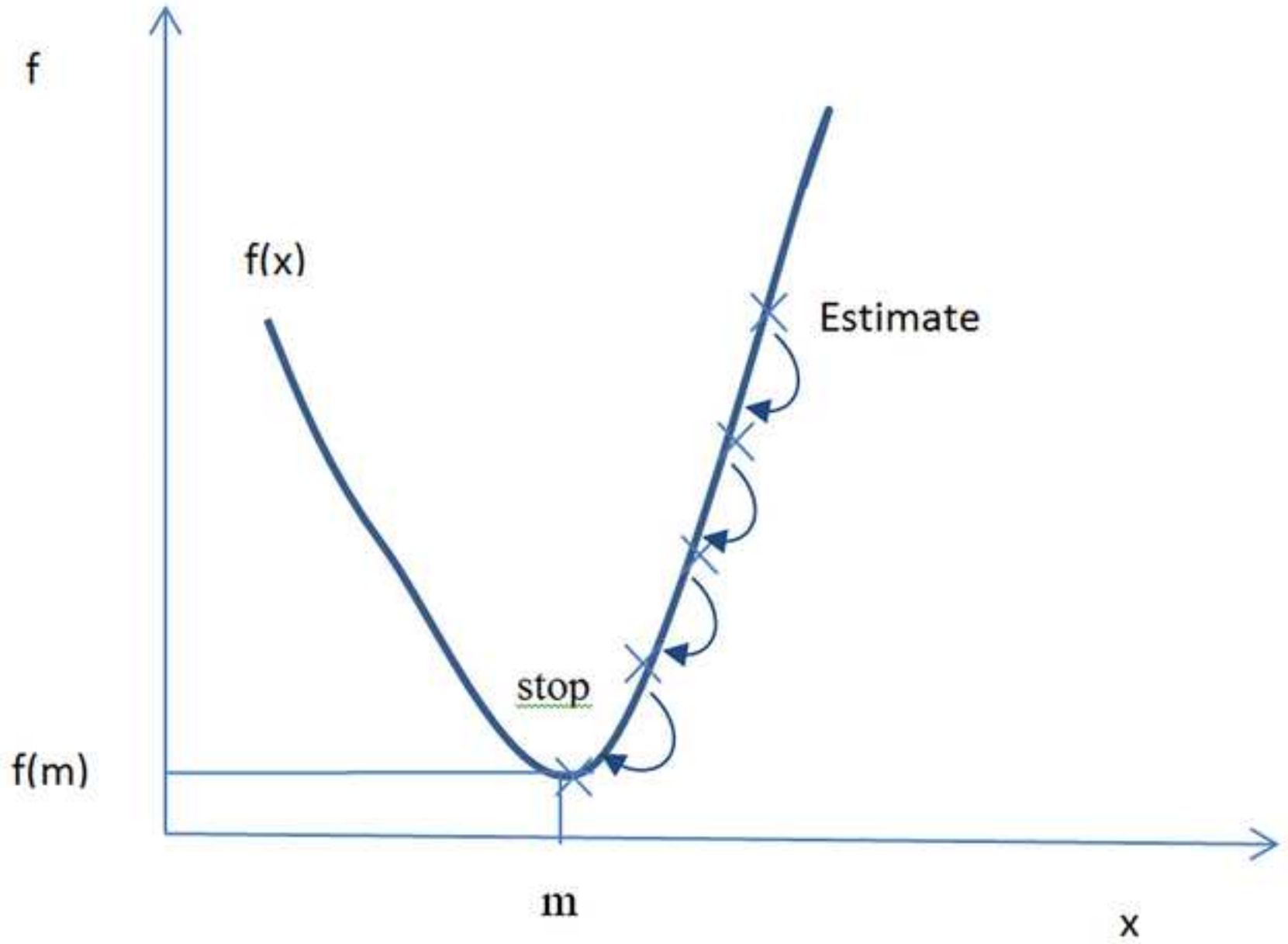
general structure of MPC
Click here to download high resolution image



general strategy of MPC
Click here to download high resolution image

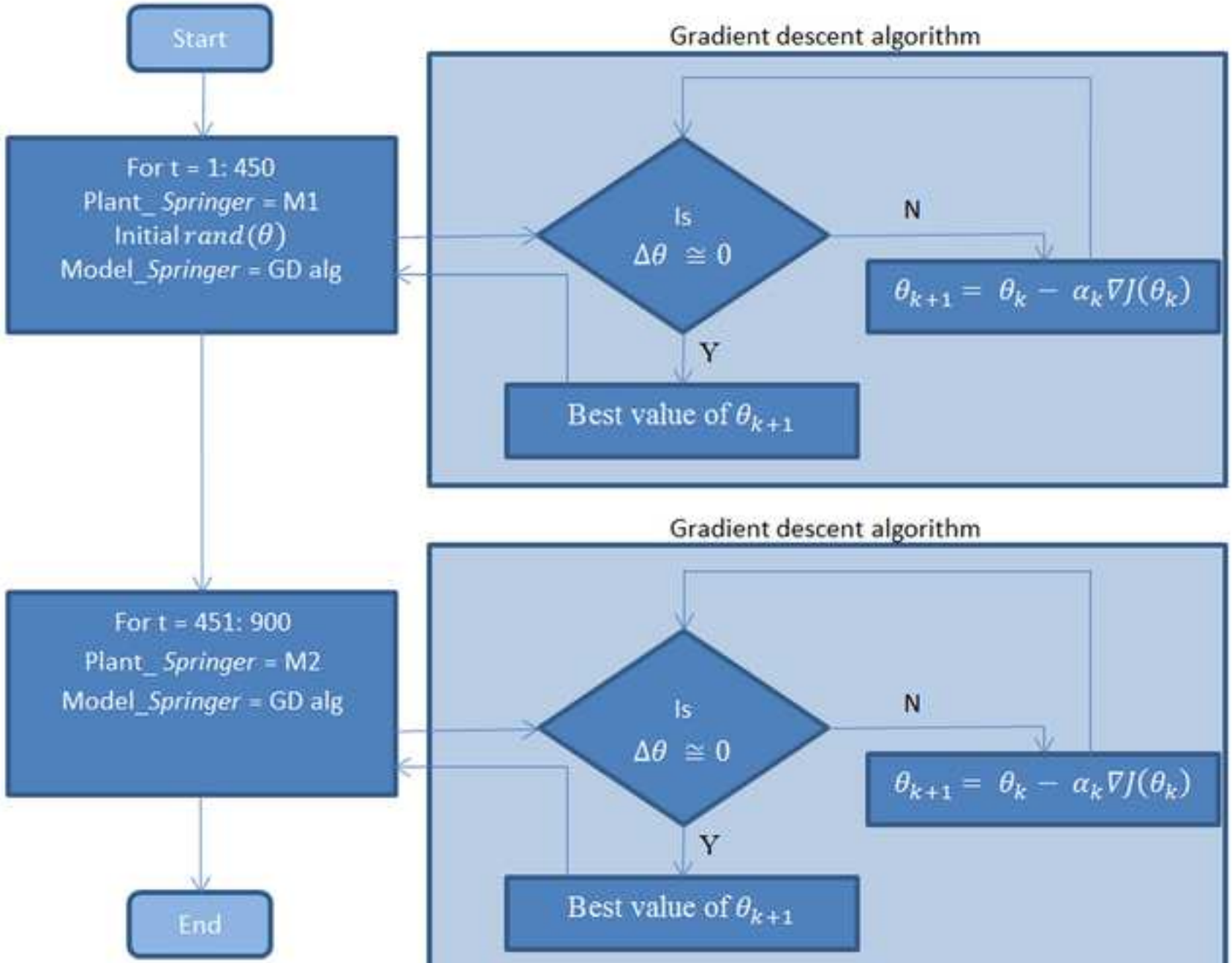


General steps involved in a gradient descent algorithm
Click here to download high resolution image

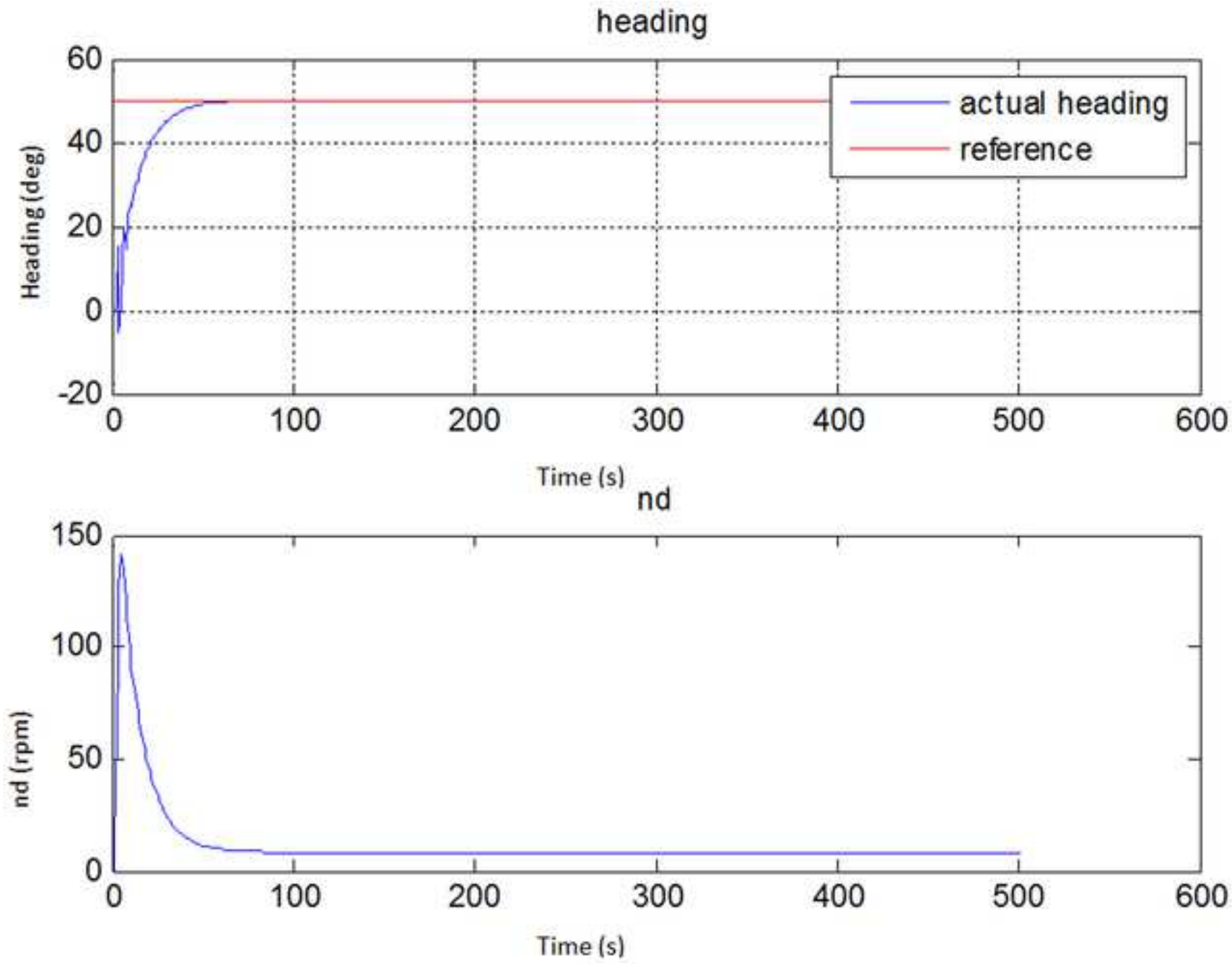


Gradient descent algorithm flow chart for Springer USV

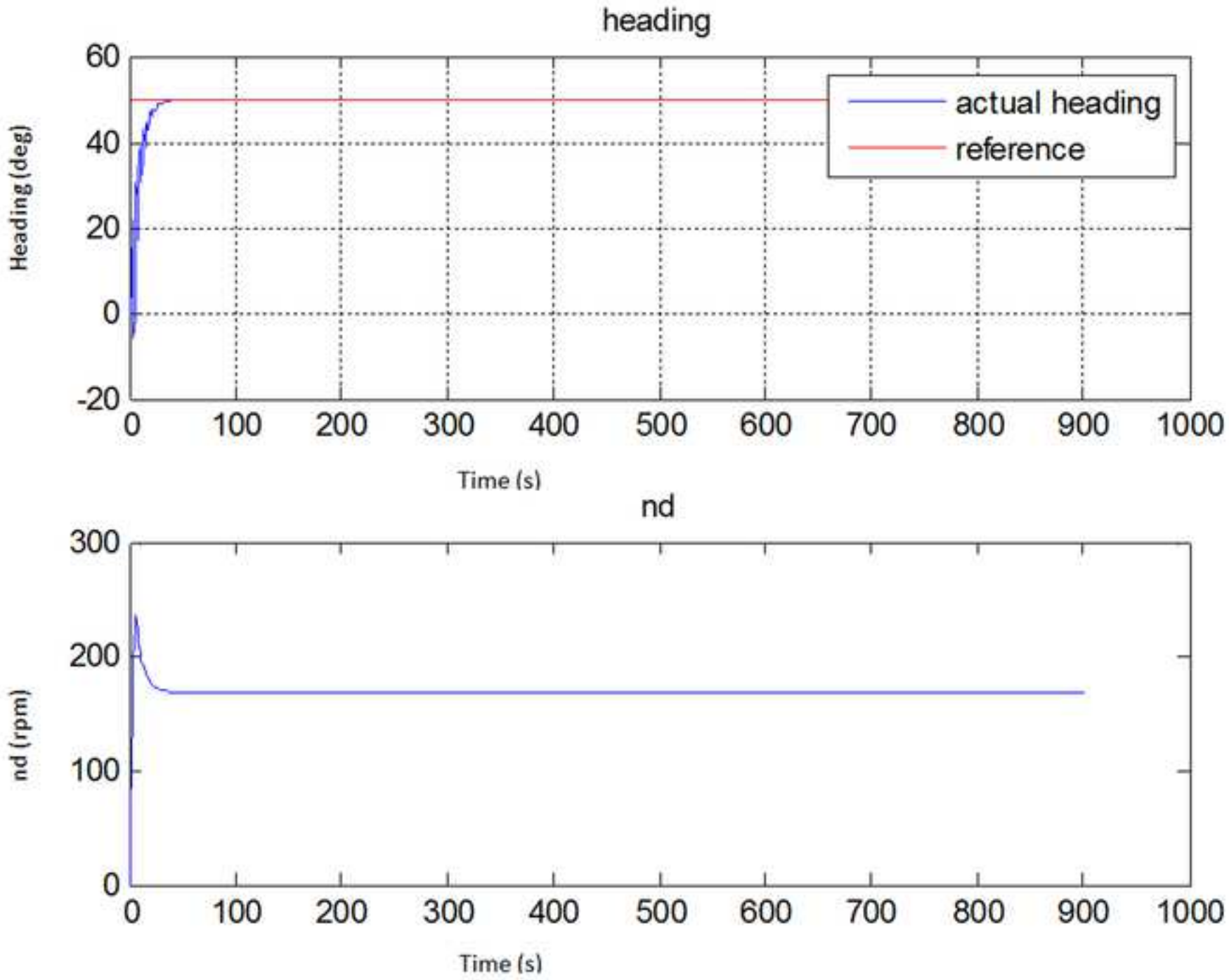
Click here to download high resolution image



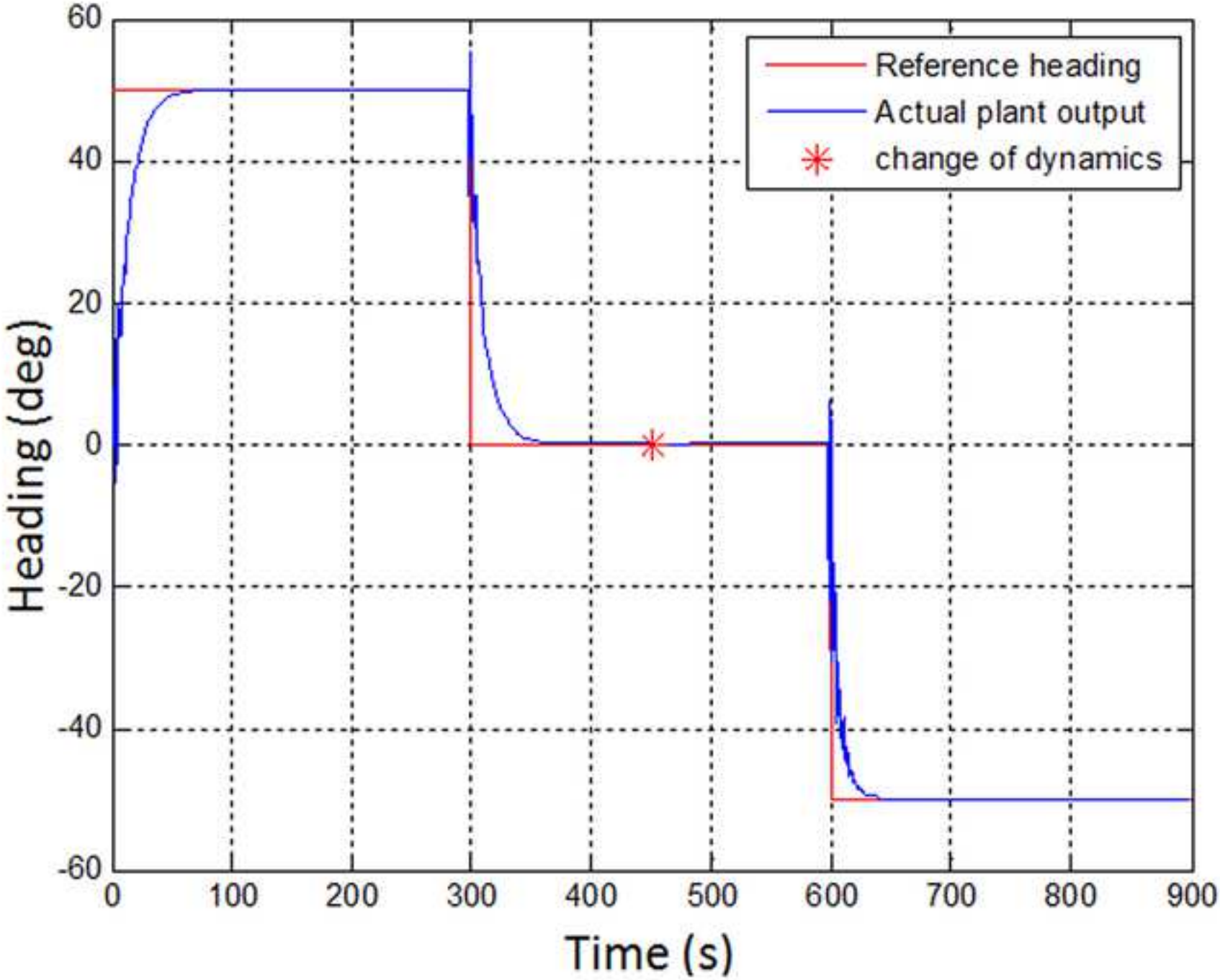
Step response with original parameters of the plant
Click here to download high resolution image



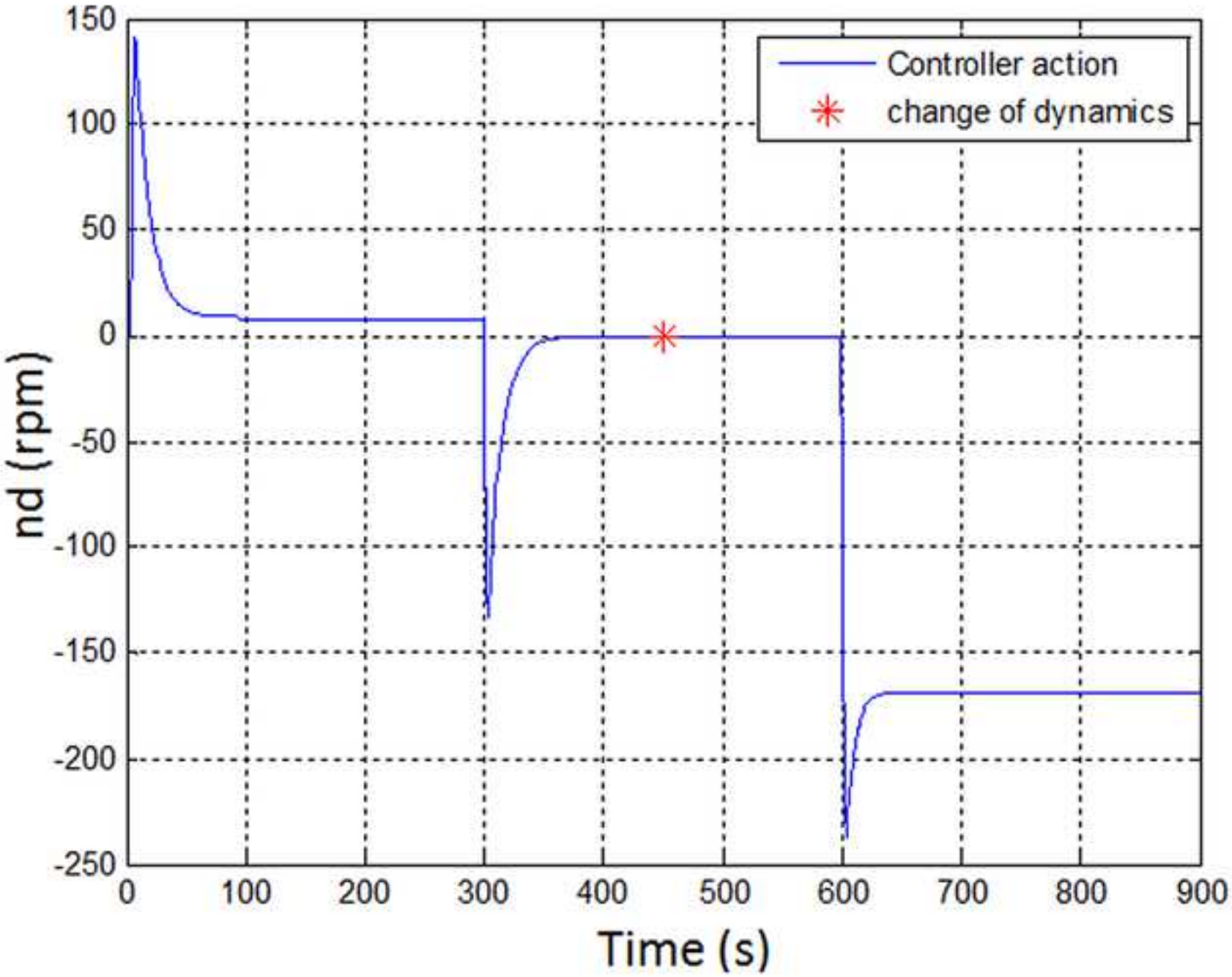
Step response for the new system dynamics of the plant
Click here to download high resolution image



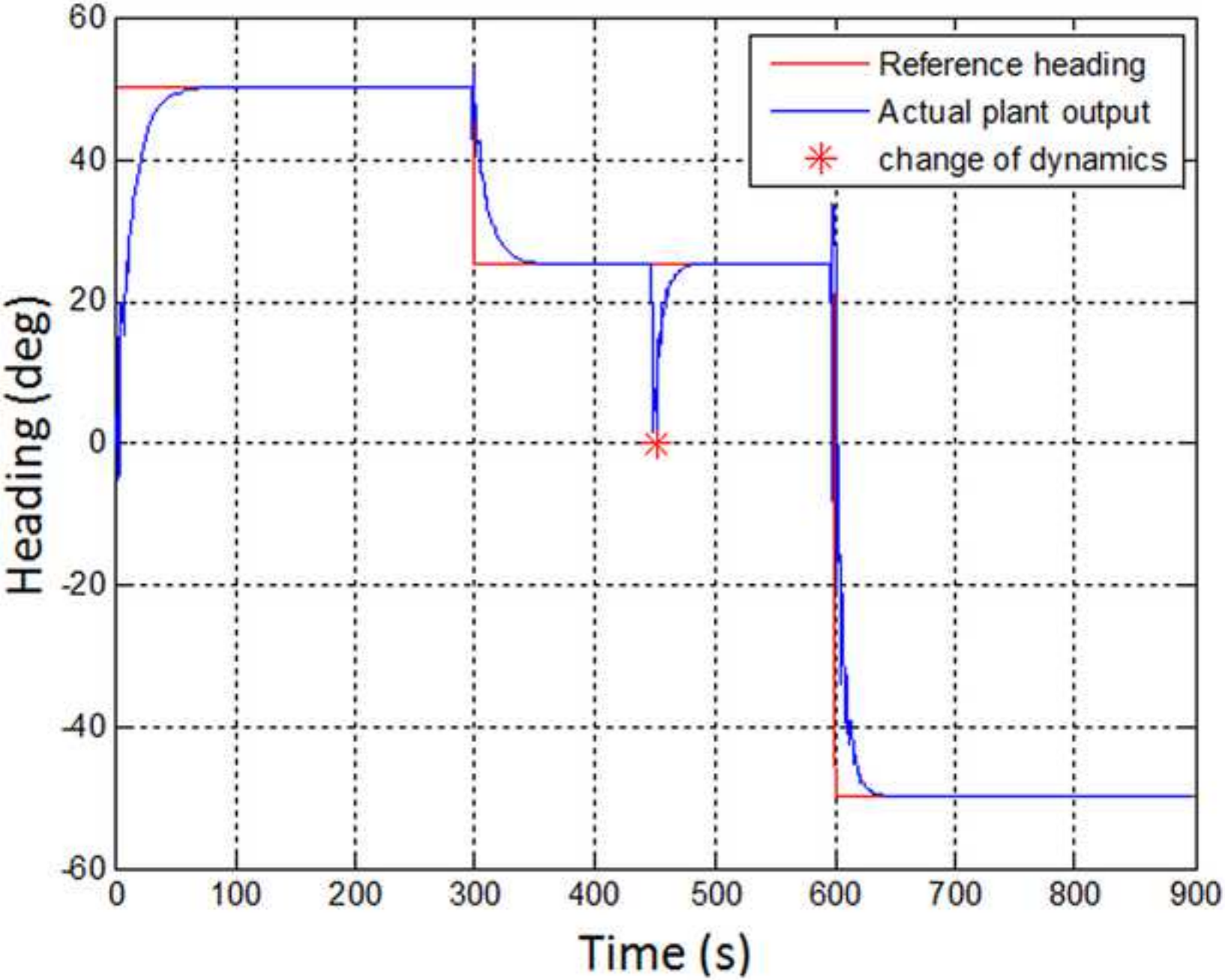
plant output and three reference signals
Click here to download high resolution image



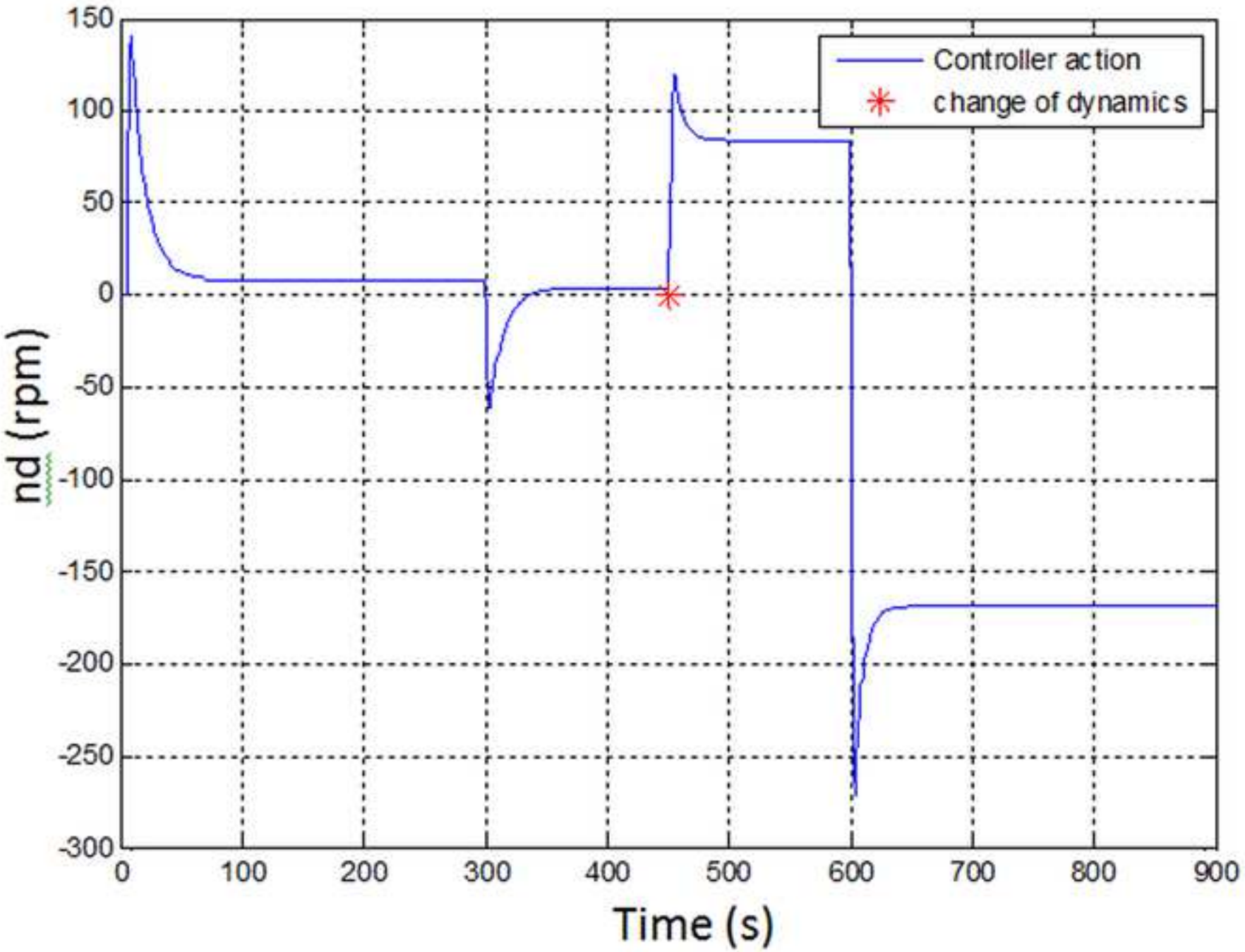
controller action
Click here to download high resolution image



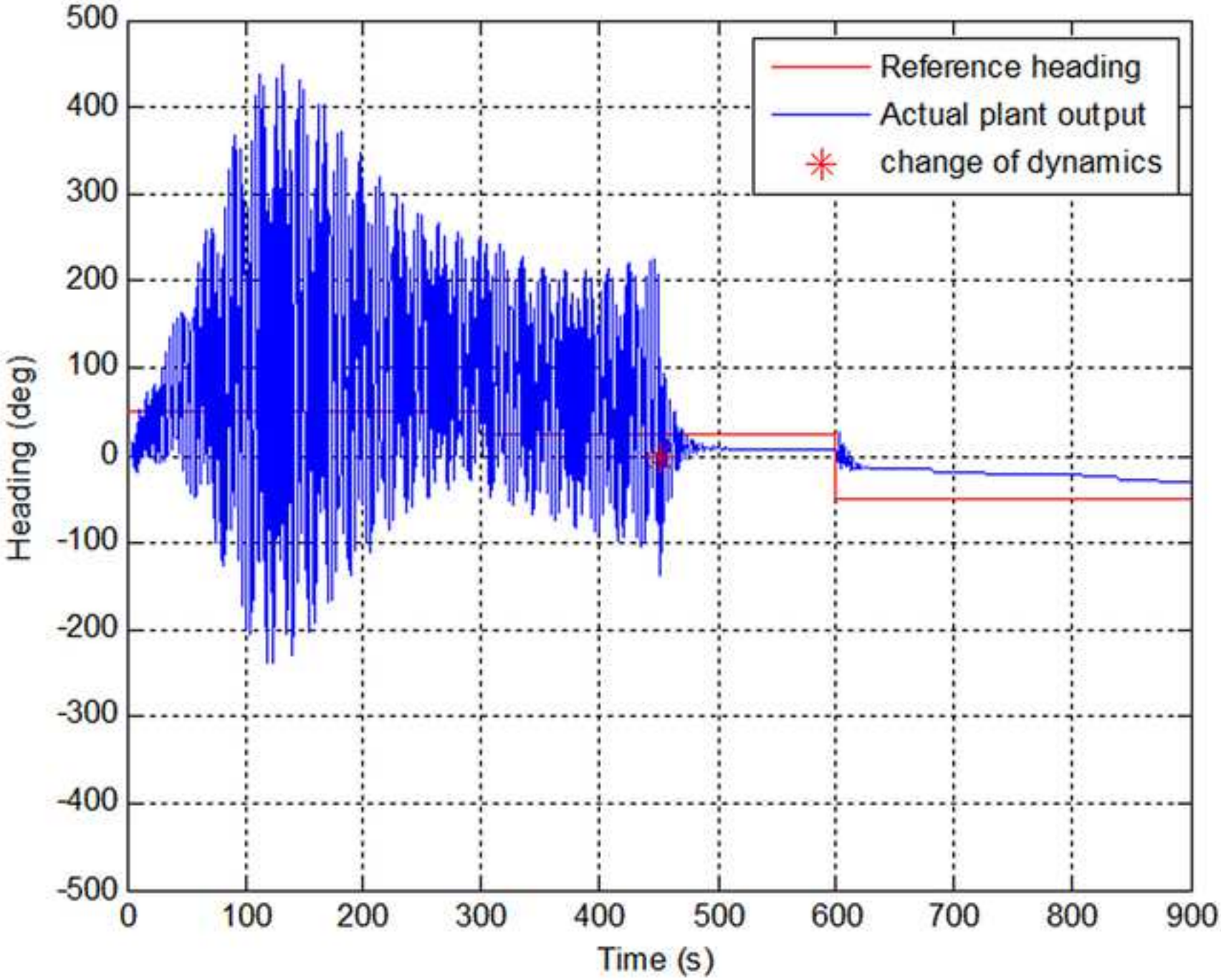
plant output
Click here to download high resolution image



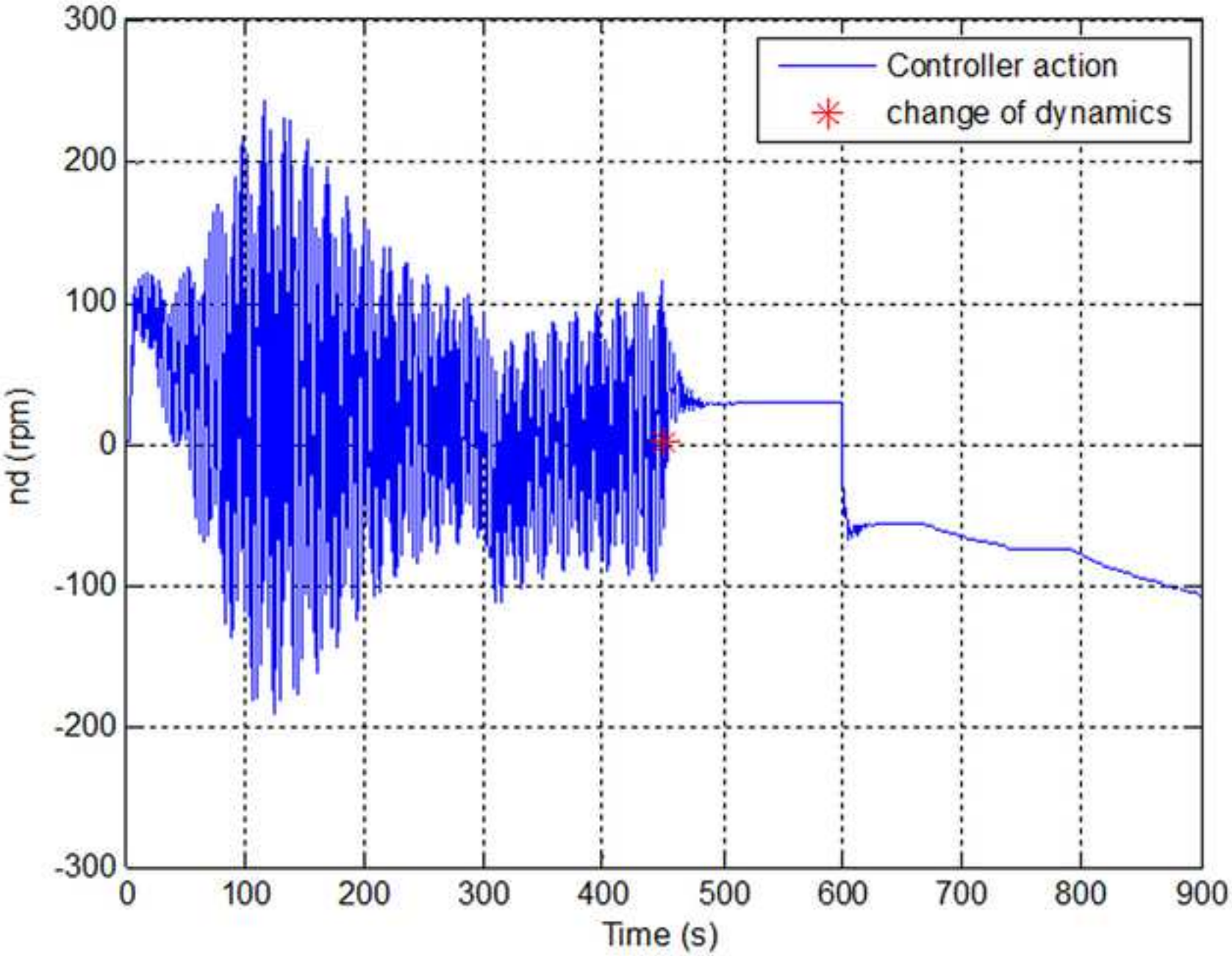
controller action
Click here to download high resolution image



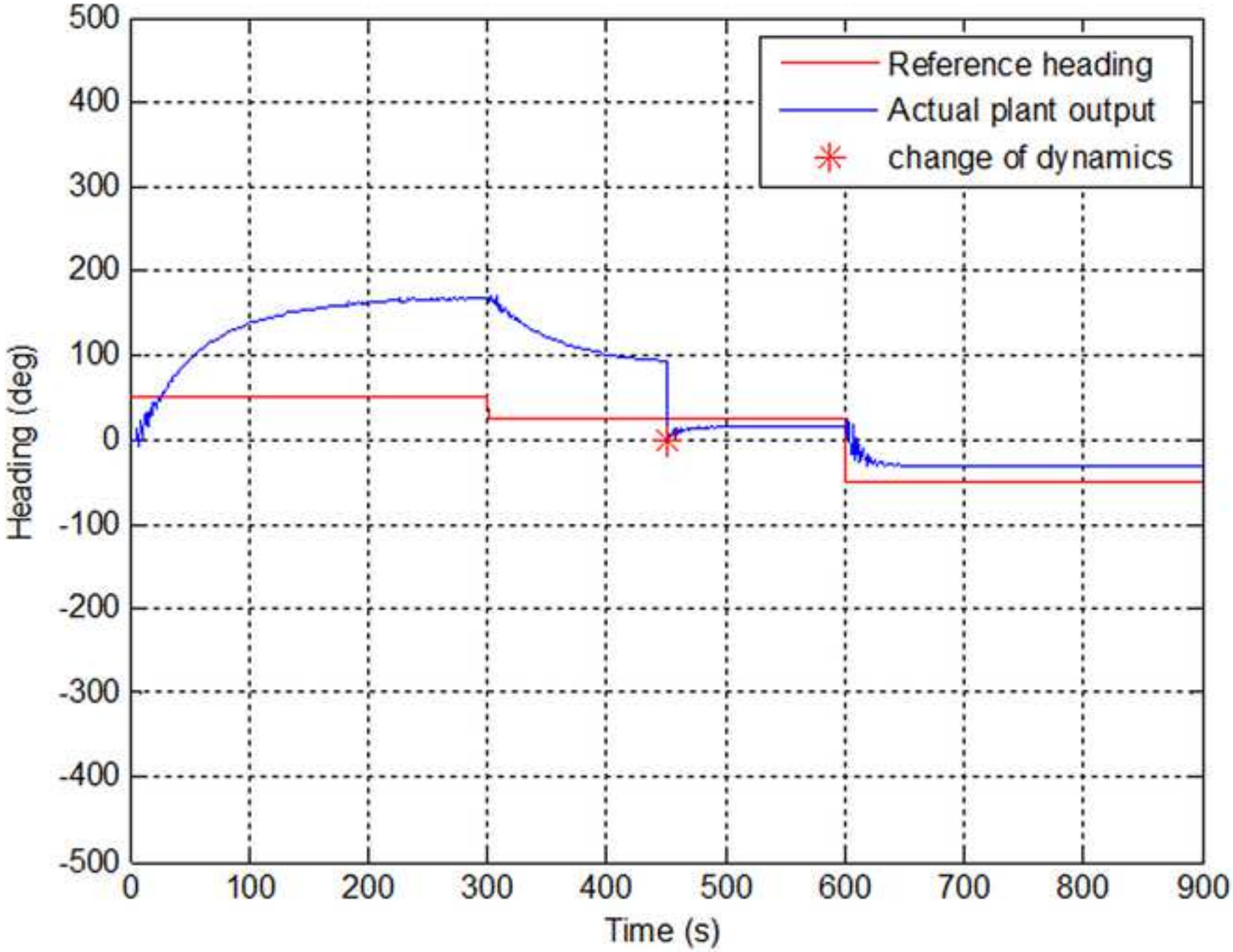
case 1; Plant output
Click here to download high resolution image



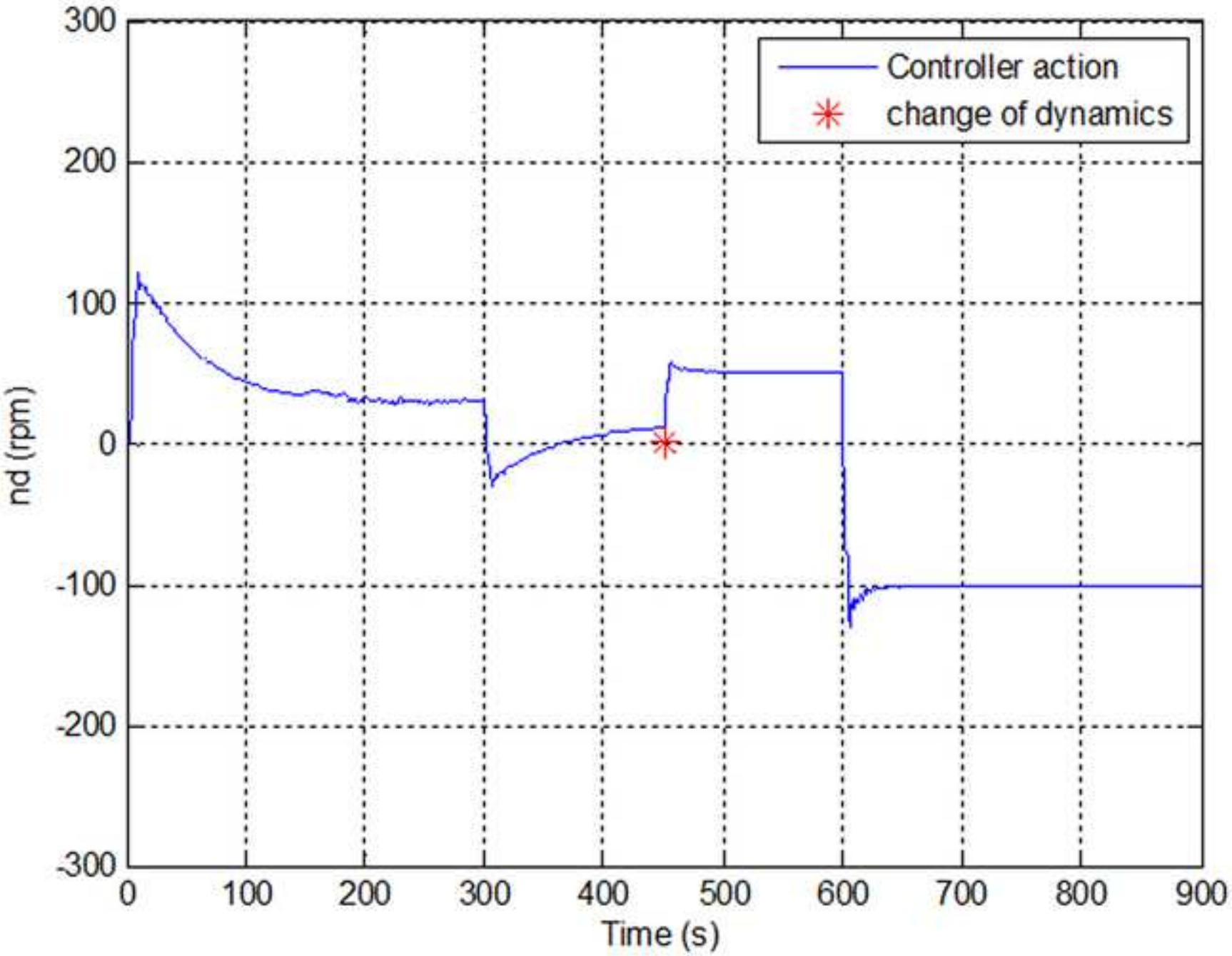
case 1; Controller action
Click here to download high resolution image



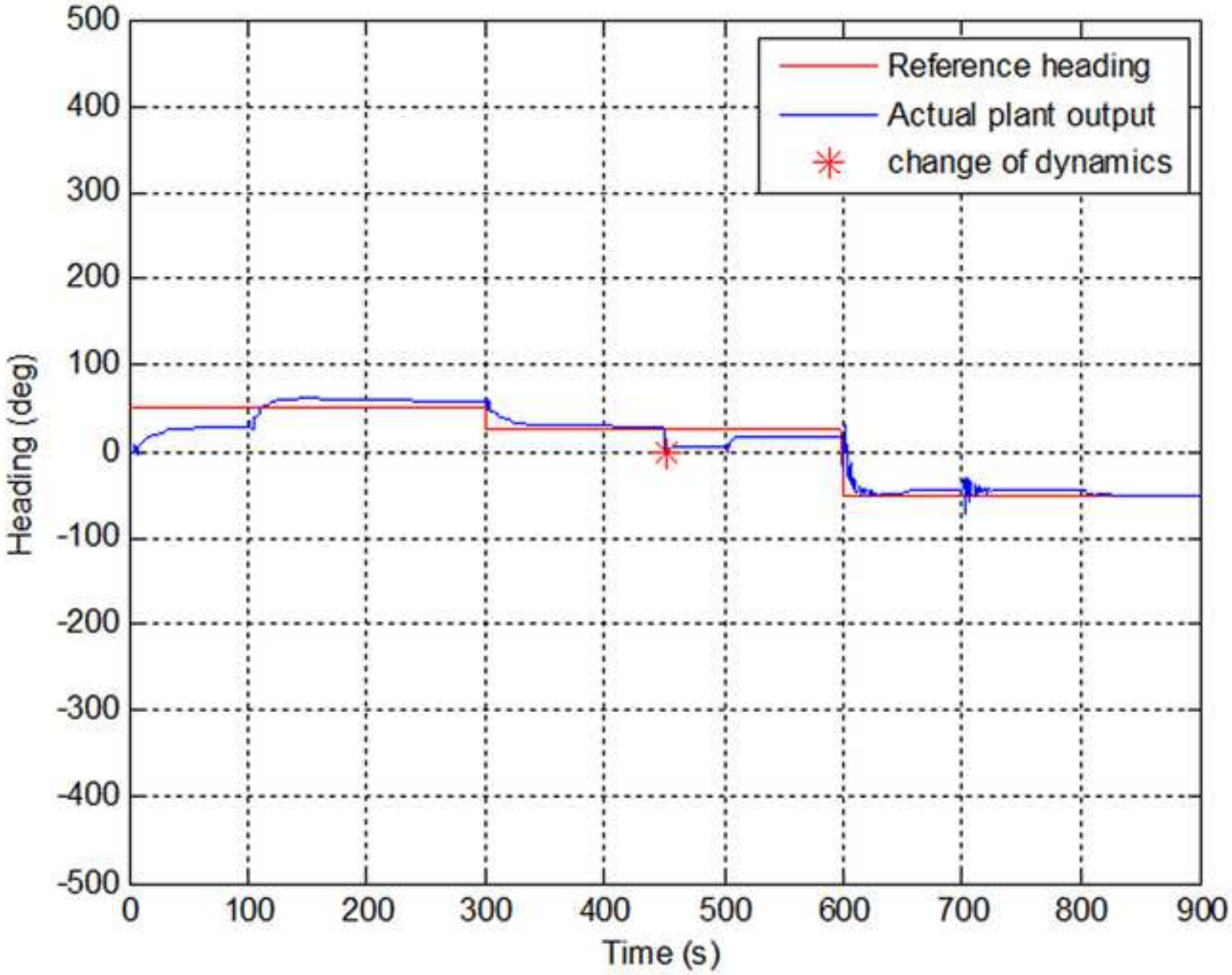
case 2; Plant output
Click here to download high resolution image



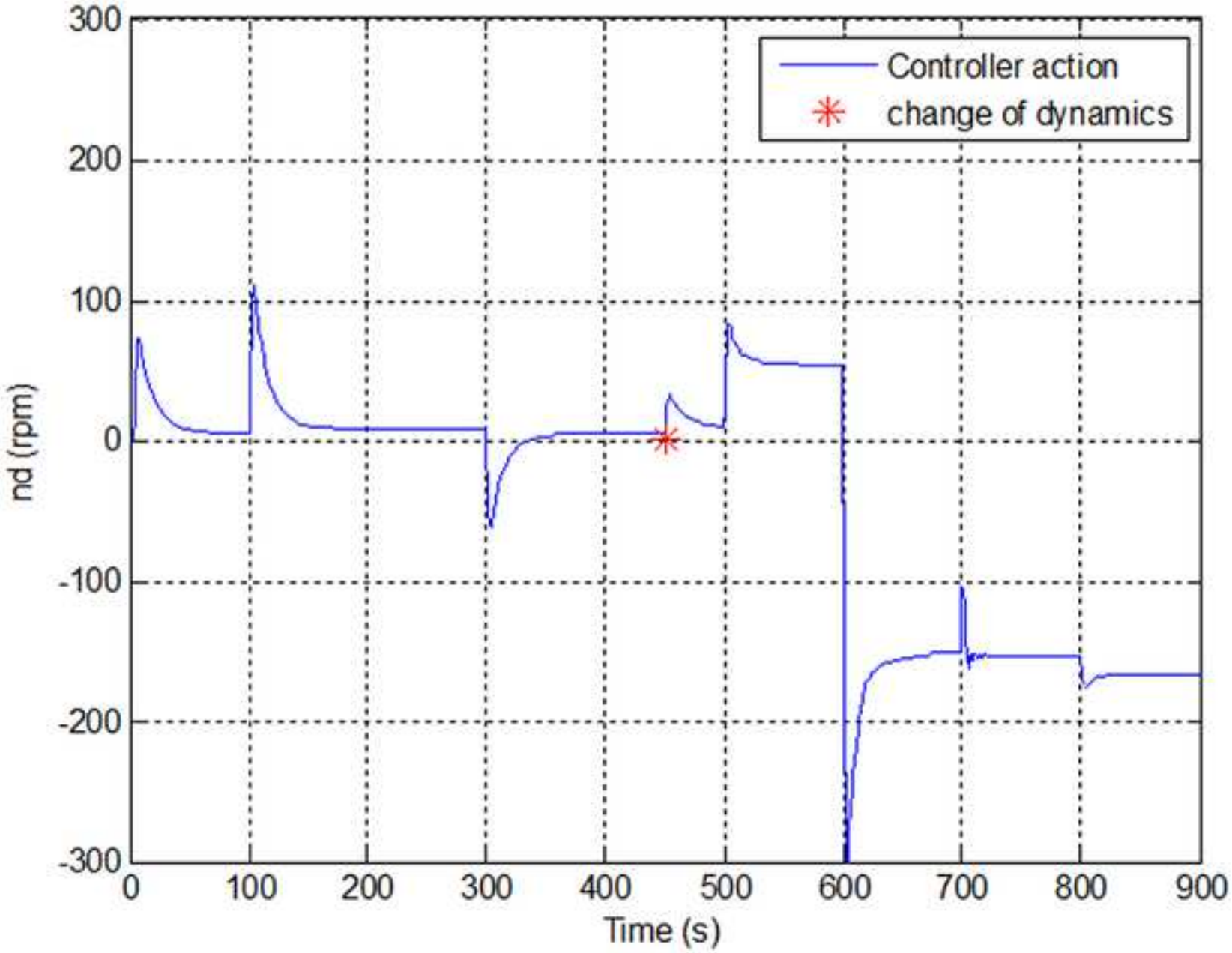
case 2; Controller action
Click here to download high resolution image



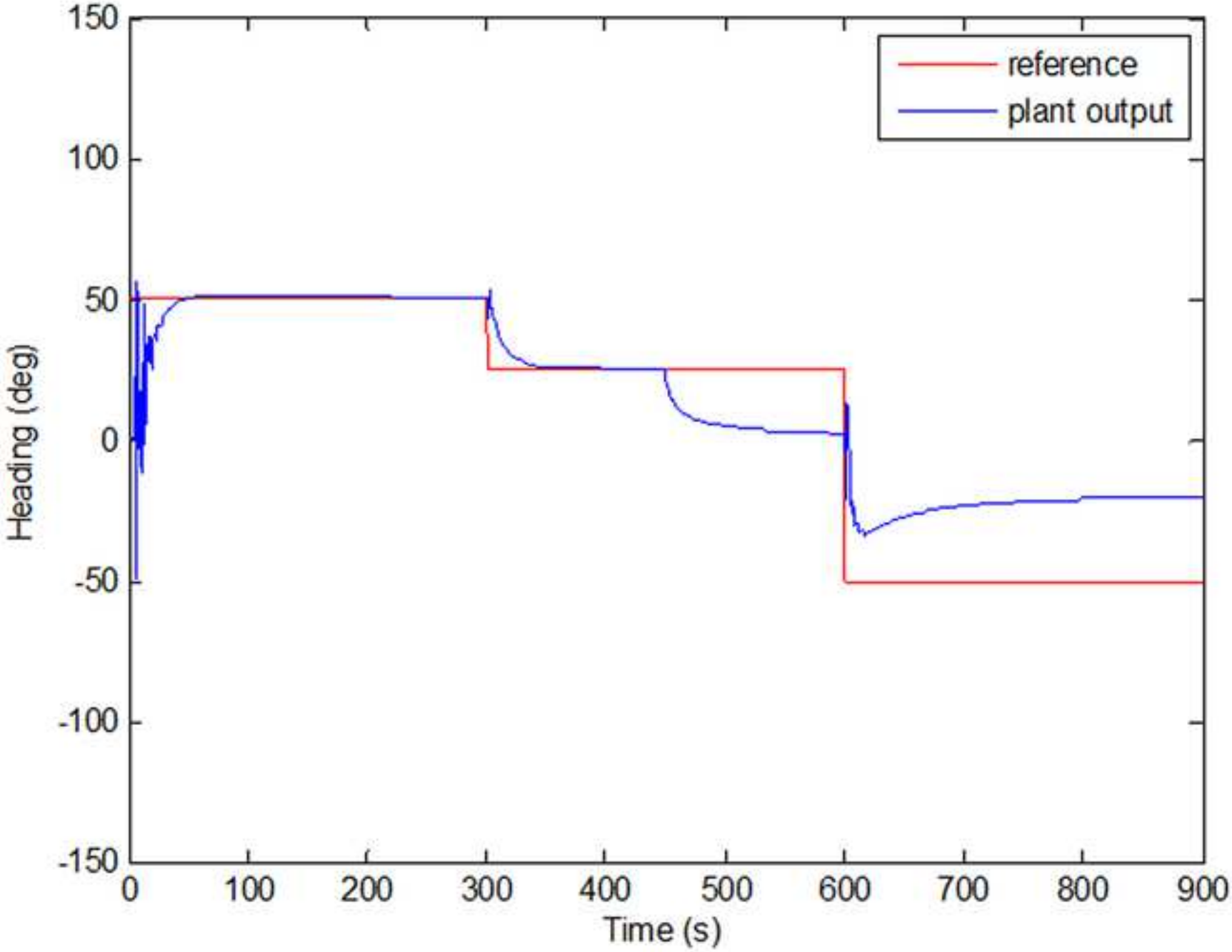
case 3; Plant output
Click here to download high resolution image



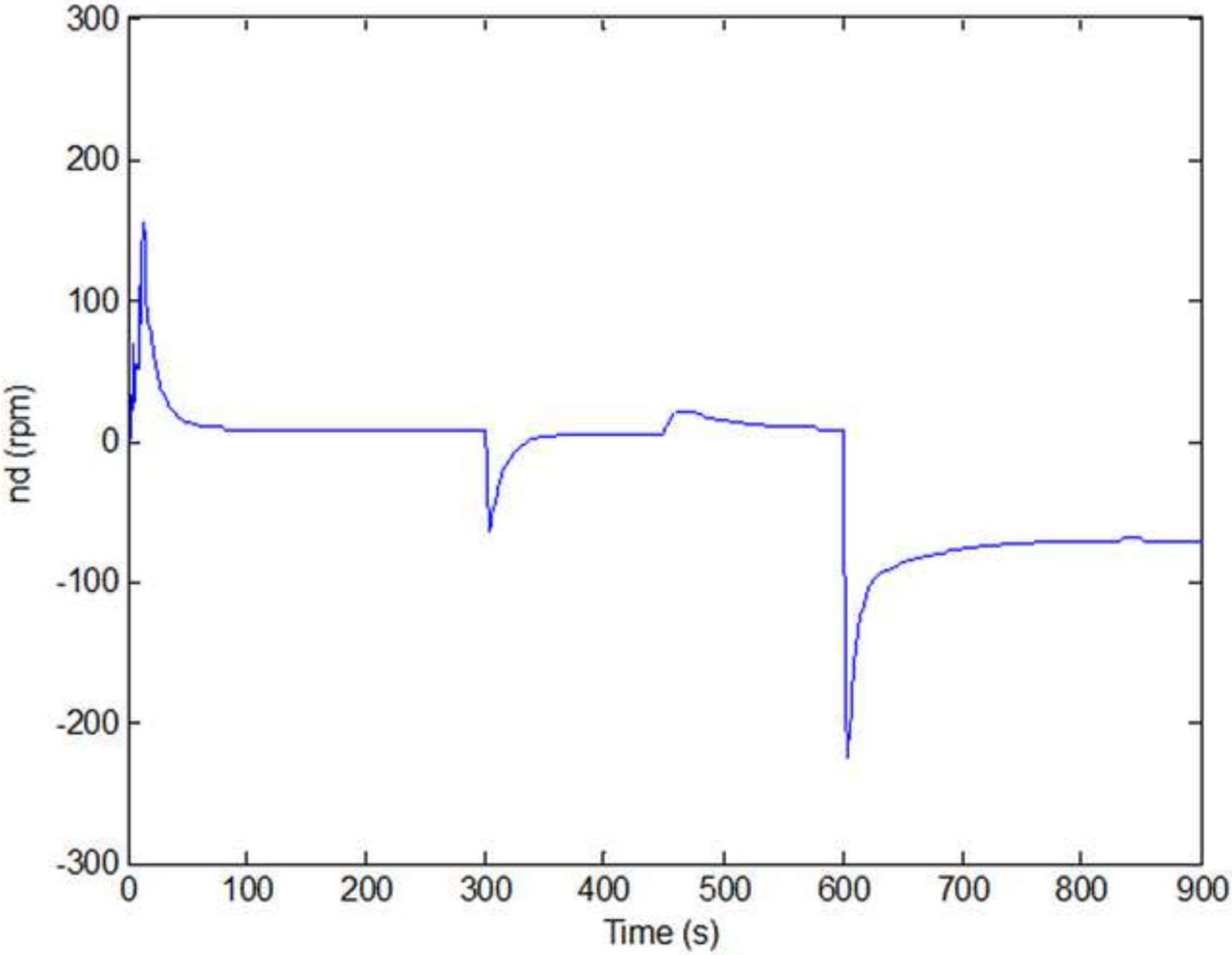
case 3; Controller action
Click here to download high resolution image



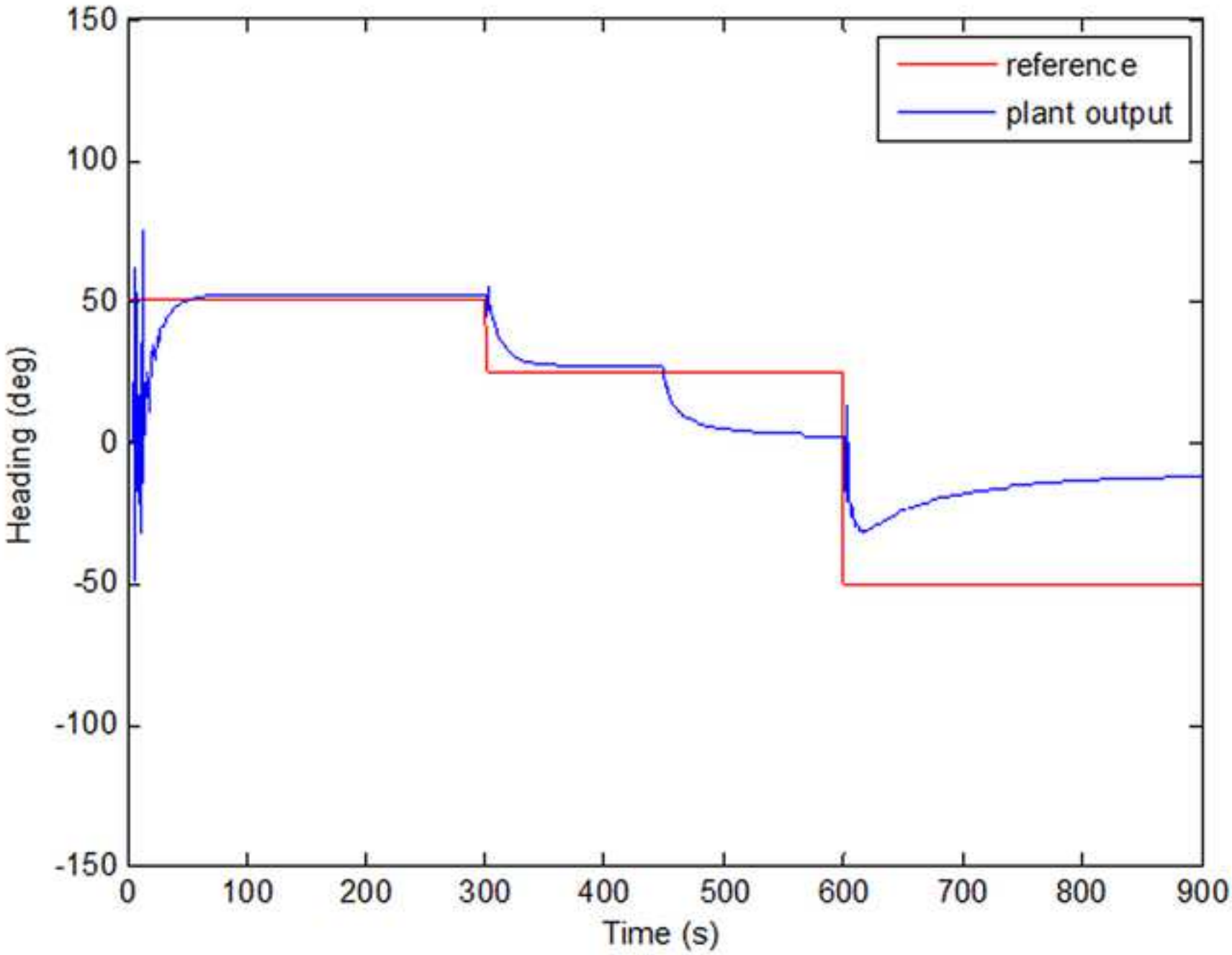
LS : plant output
Click here to download high resolution image



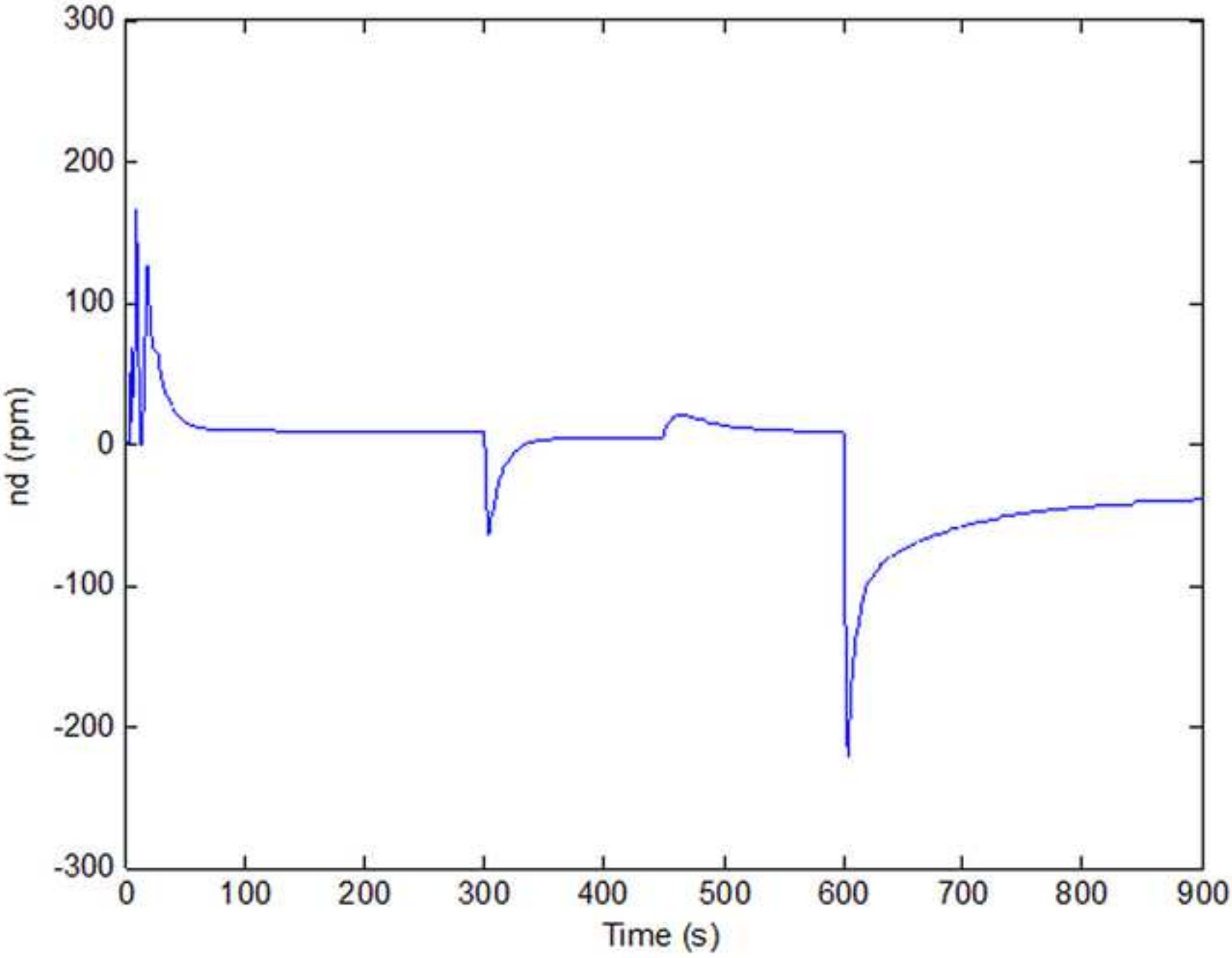
LS : controller action
Click here to download high resolution image



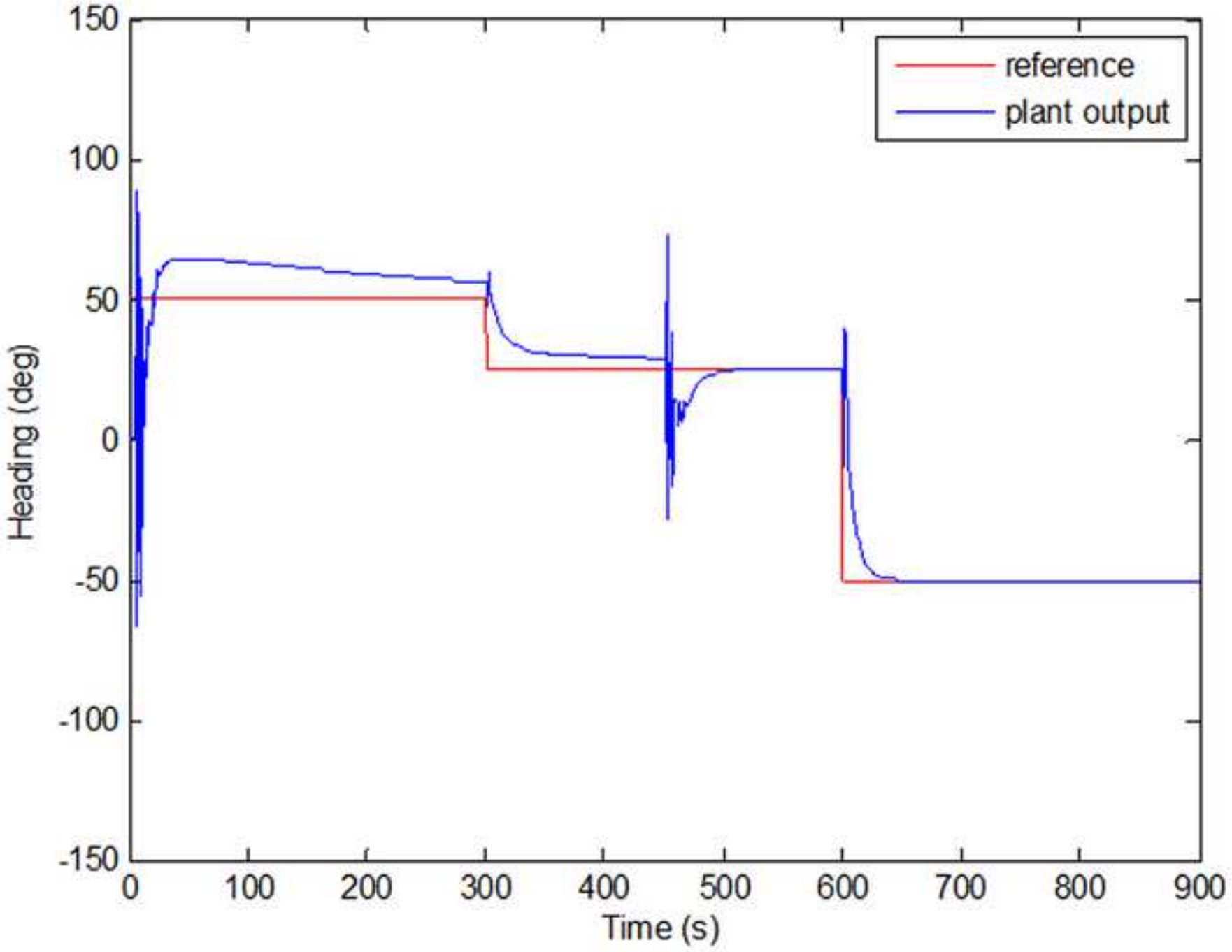
WLS : plant output
Click here to download high resolution image



WLS : controller action
Click here to download high resolution image

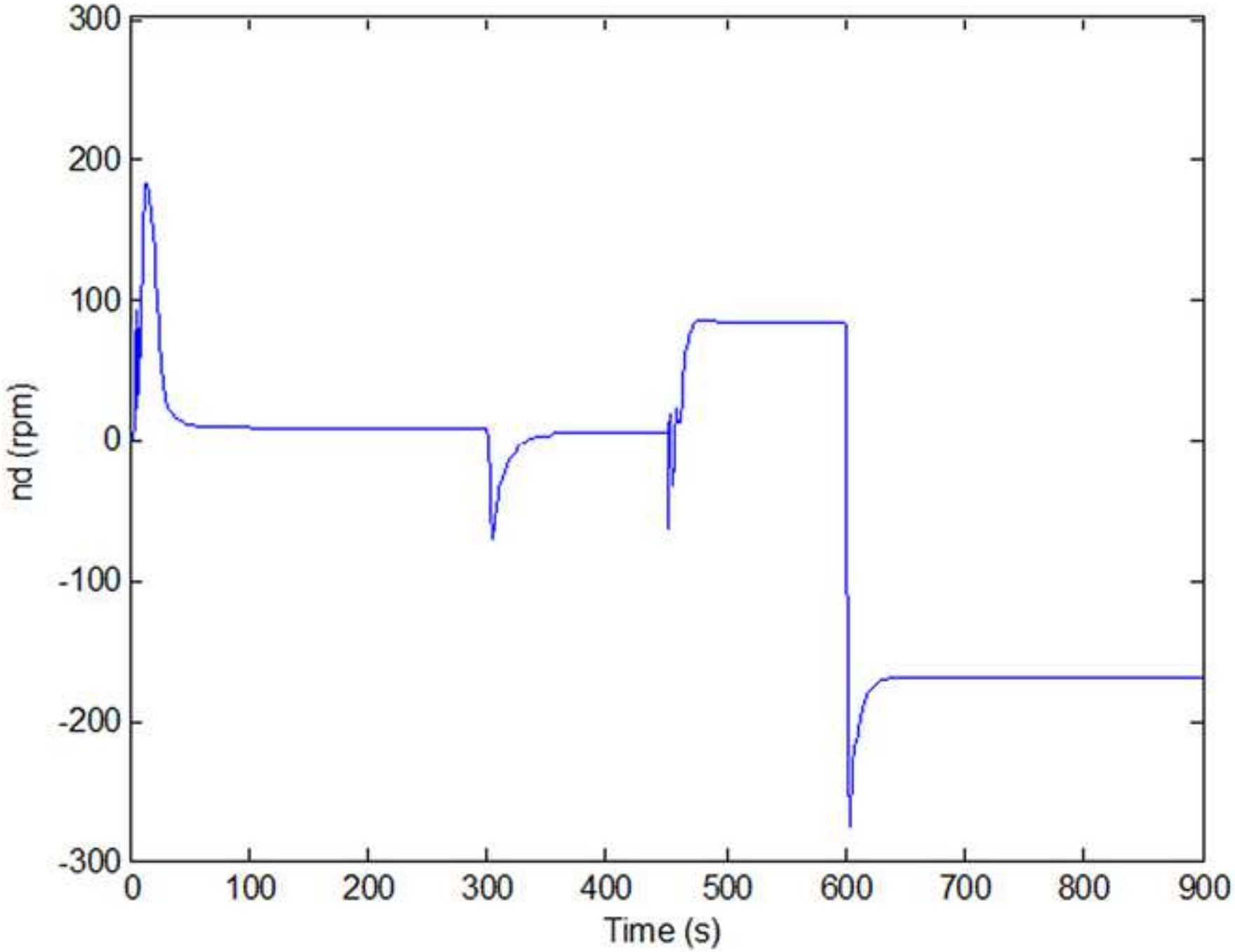


WLS re-initialised Plant output
Click here to download high resolution image

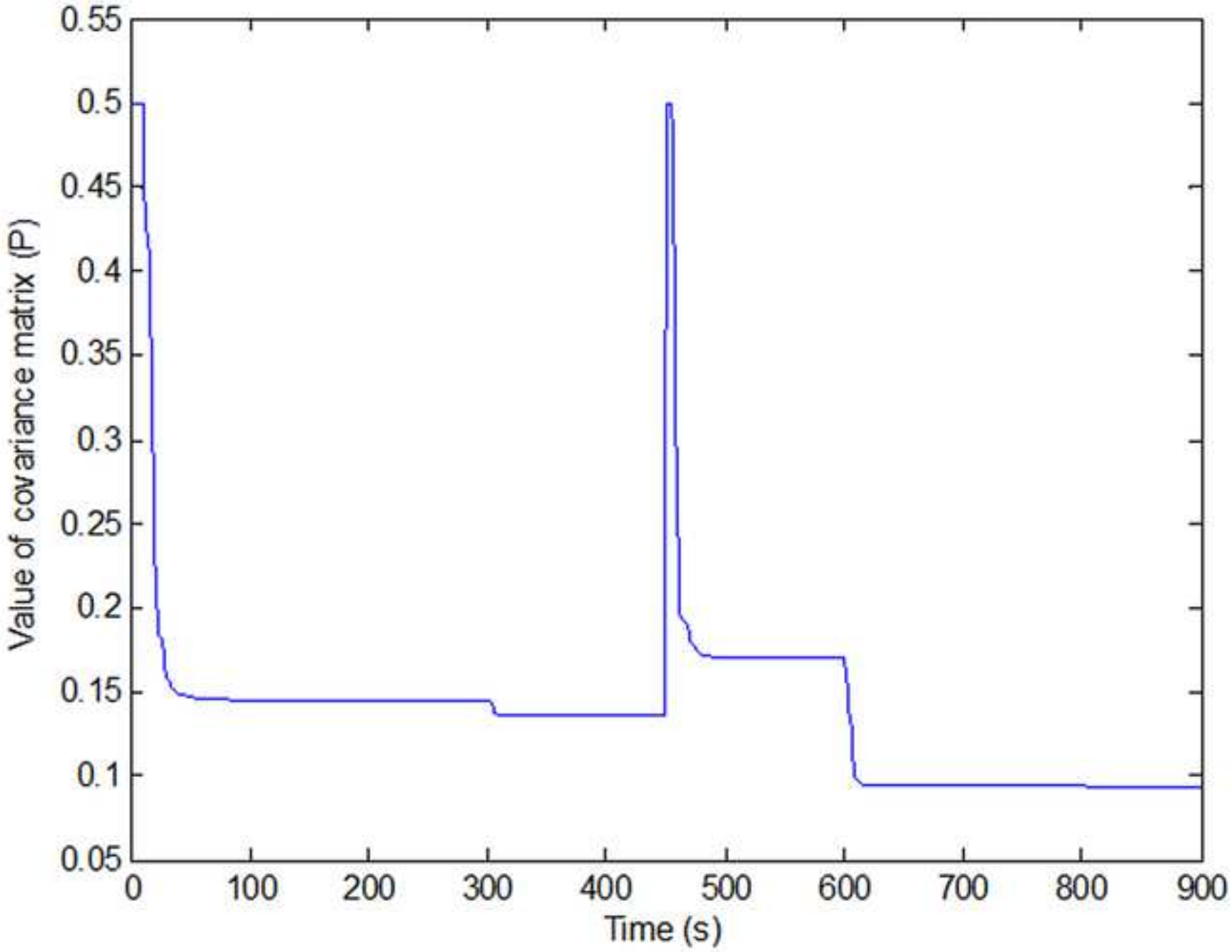


WLS re-initialised controller action

Click here to download high resolution image

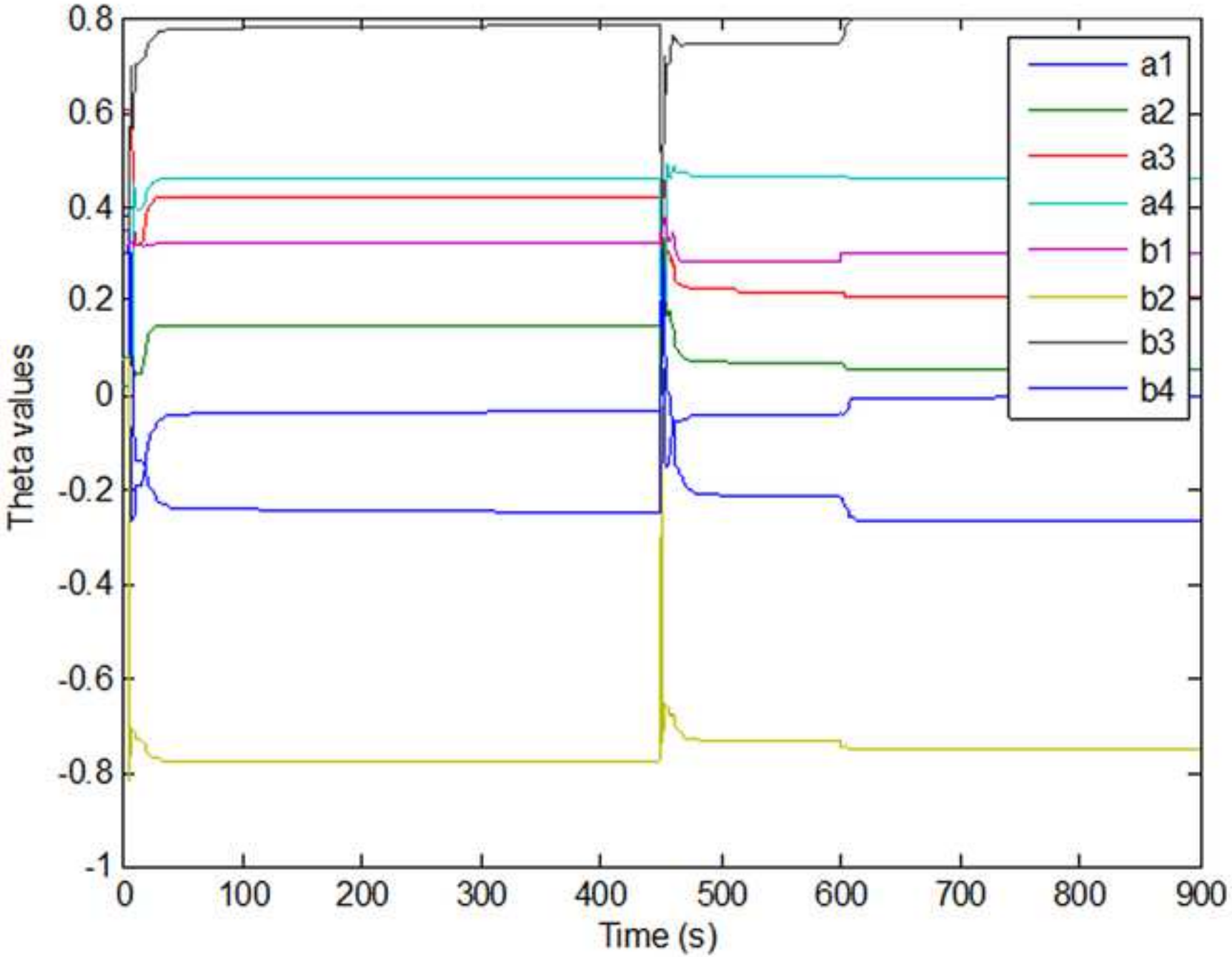


Re-initialised covariance matrix (P)
Click here to download high resolution image

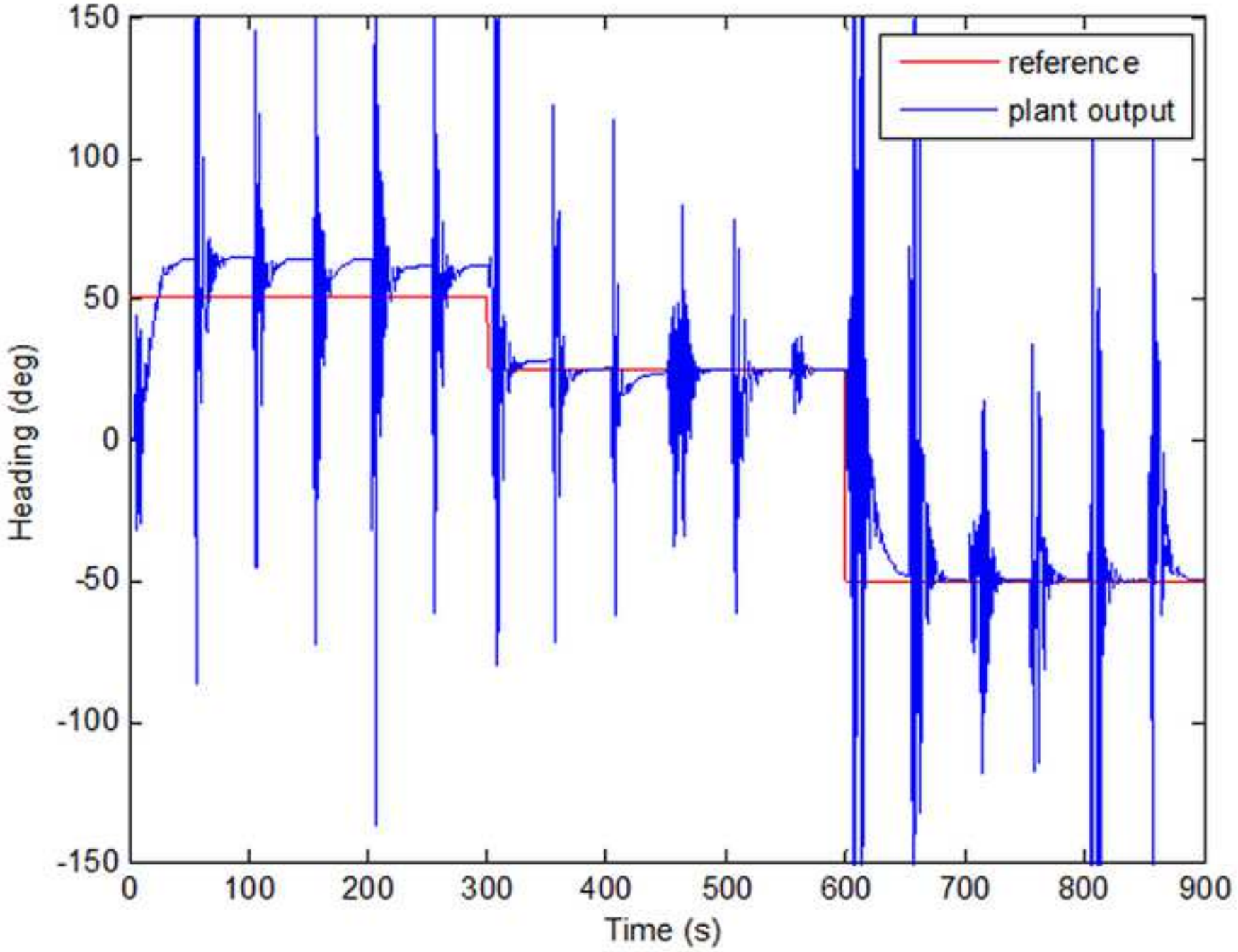


θ values

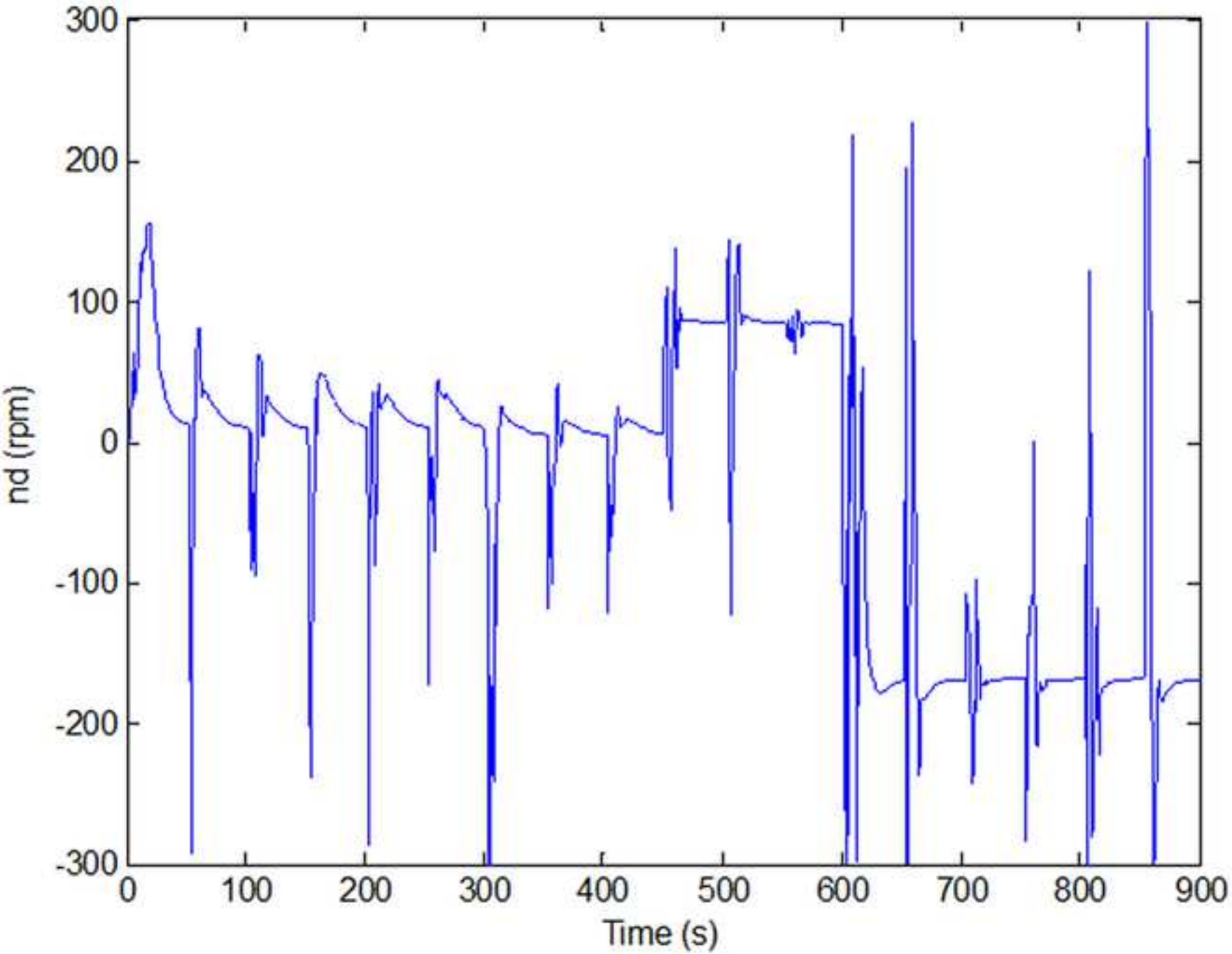
Click here to download high resolution image



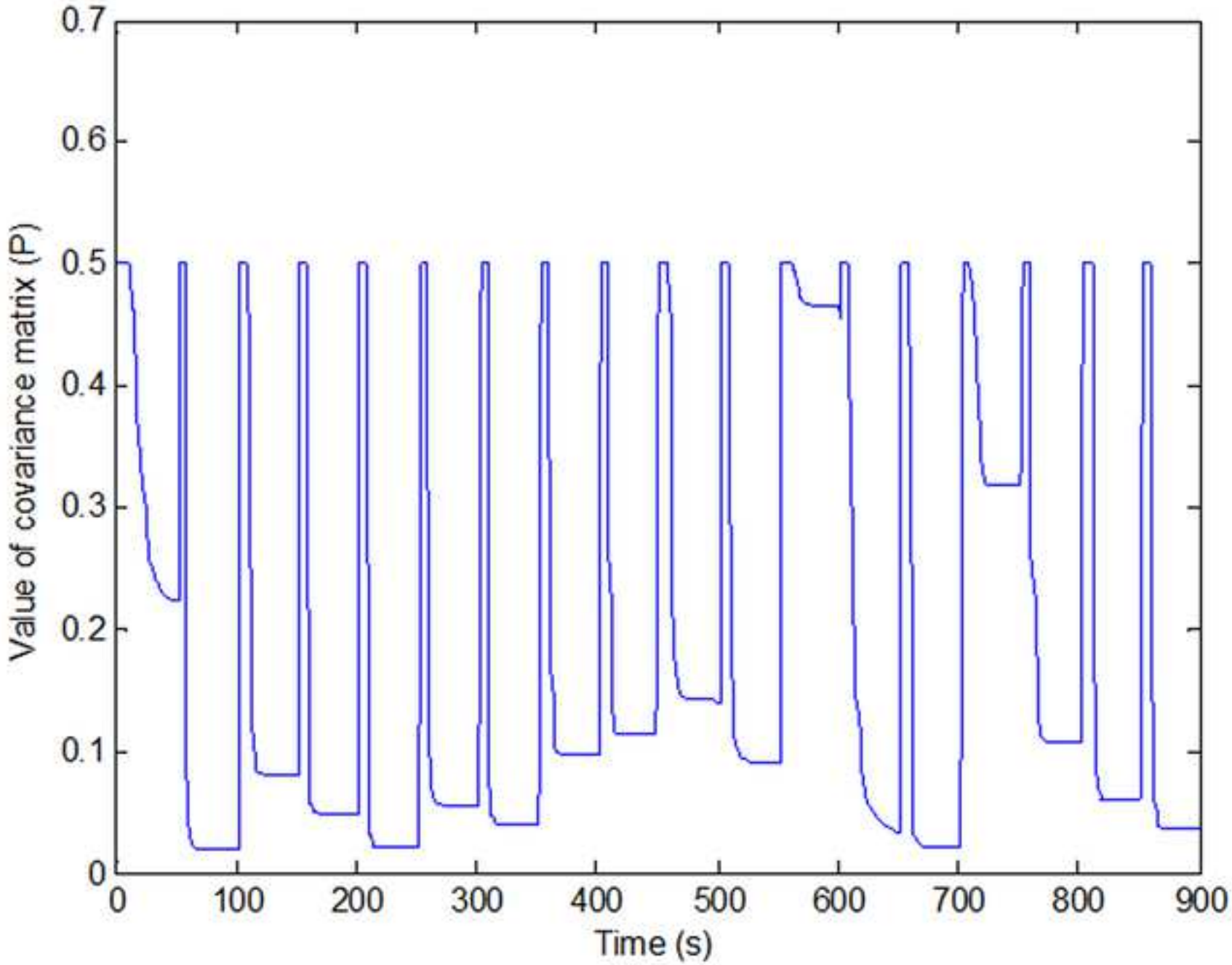
WLS re-initialised Plant output
Click here to download high resolution image



WLS re-initialised controller action
Click here to download high resolution image

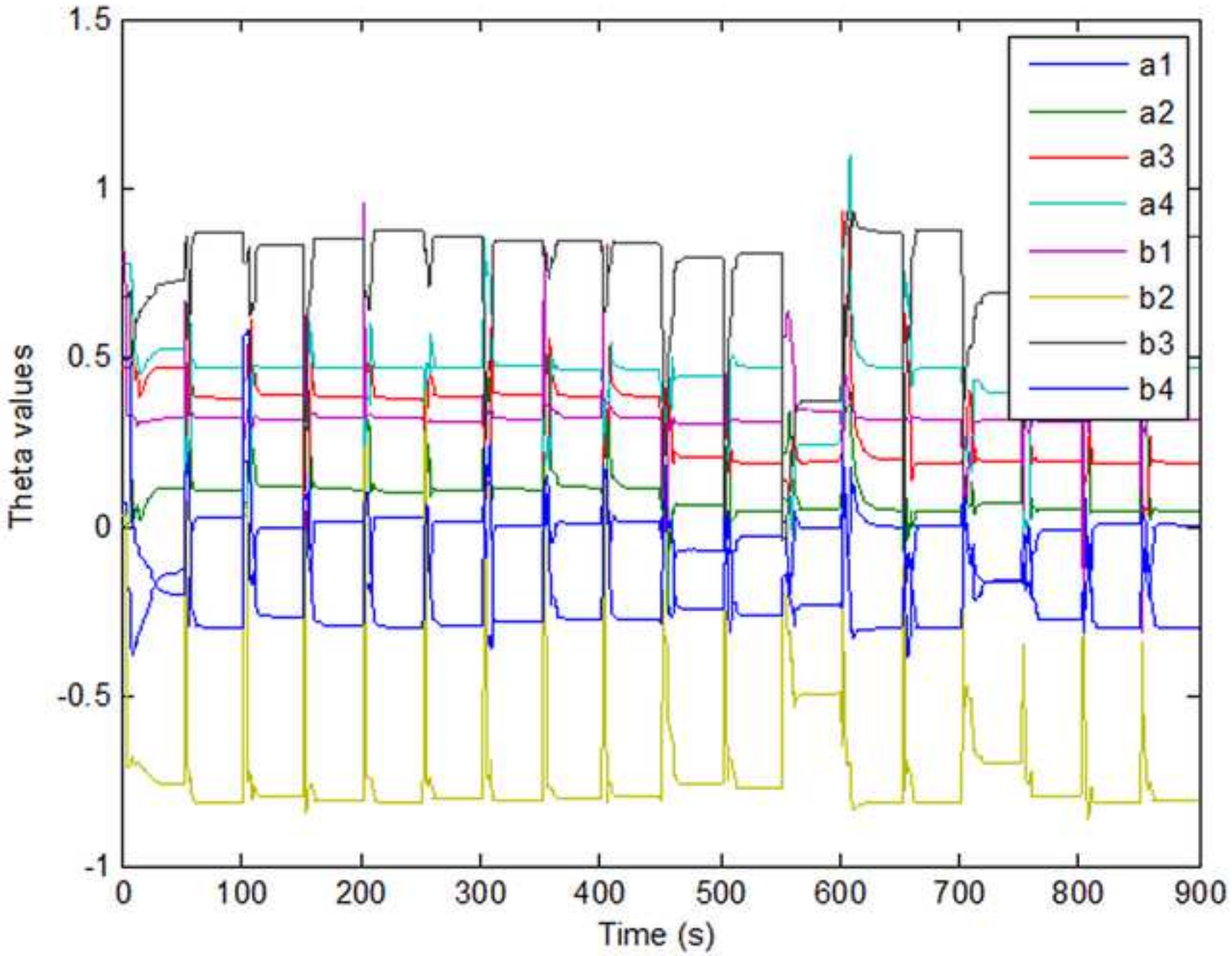


Re-initialised covariance matrix (P)
Click here to download high resolution image

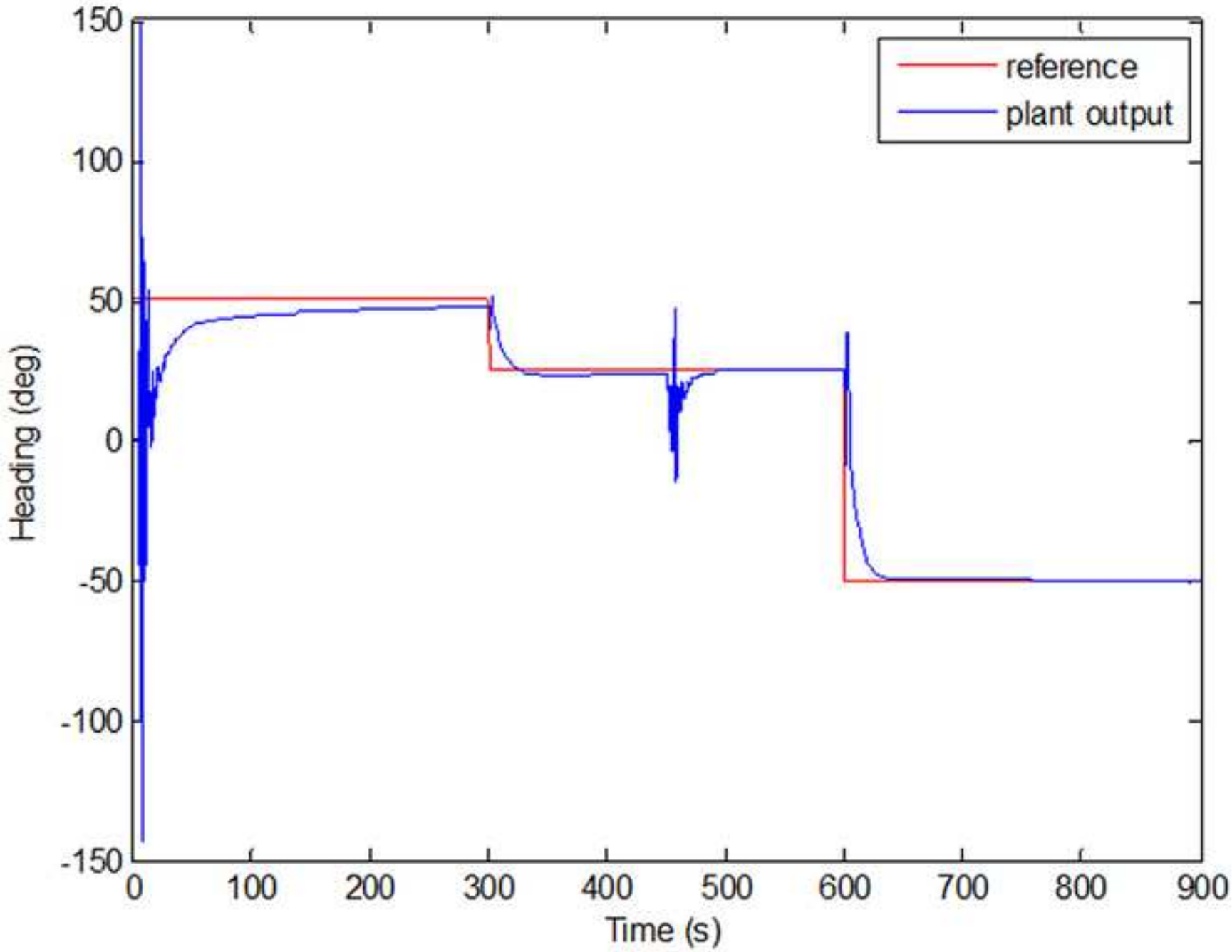


θ values

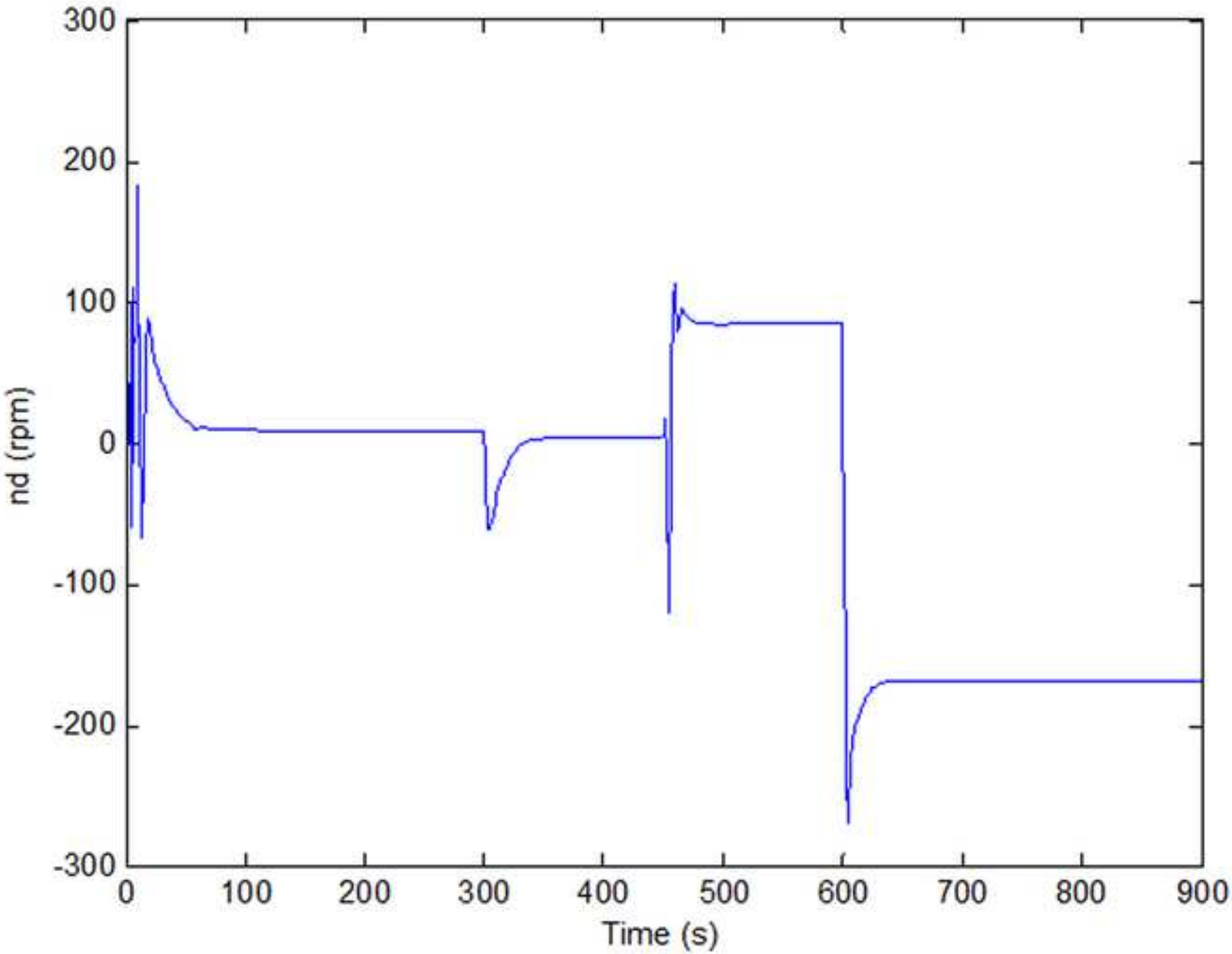
Click here to download high resolution image



(P) re-initialised every 50 s Plant output
Click here to download high resolution image

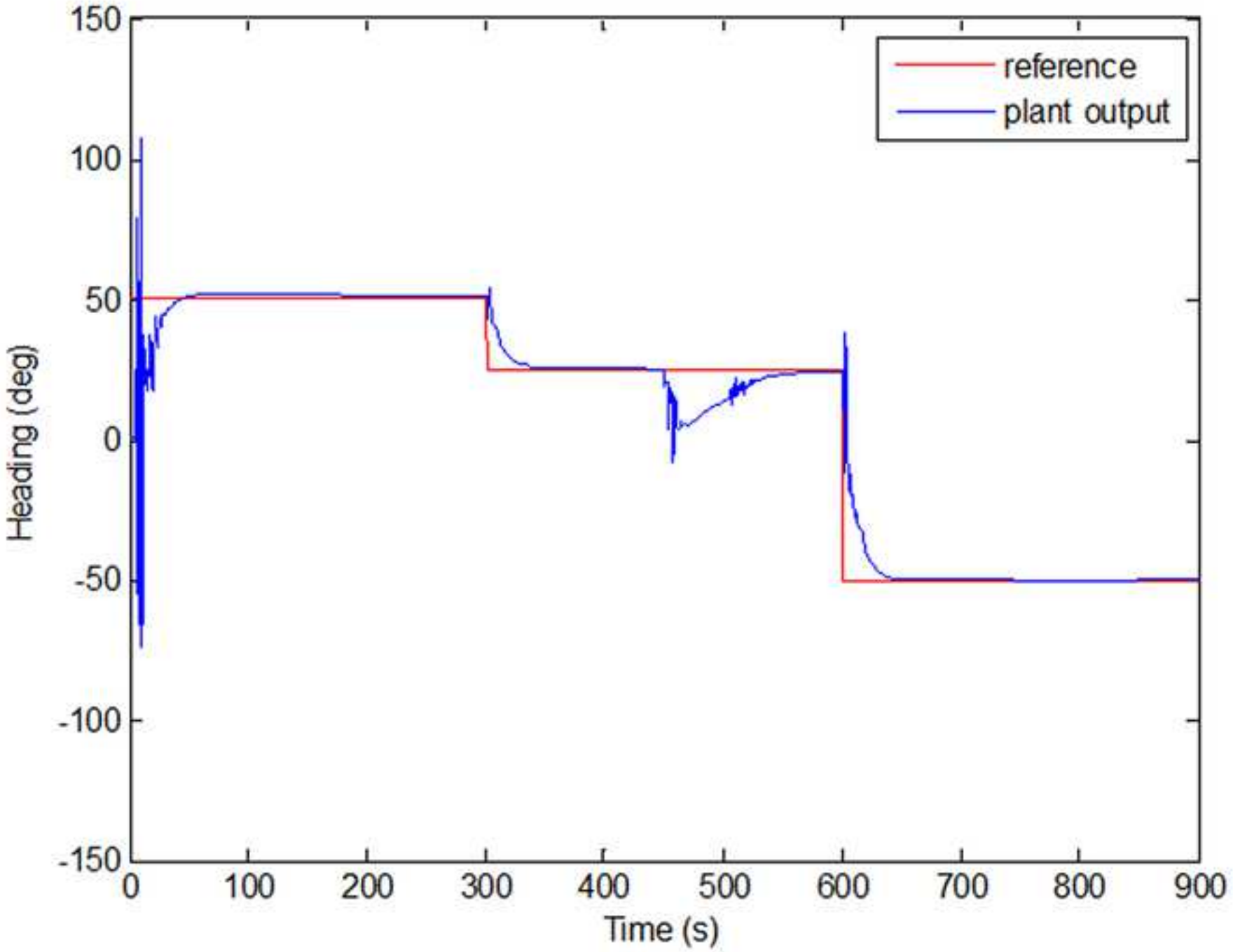


(P) re-initialised every 50 s Controller action
Click here to download high resolution image



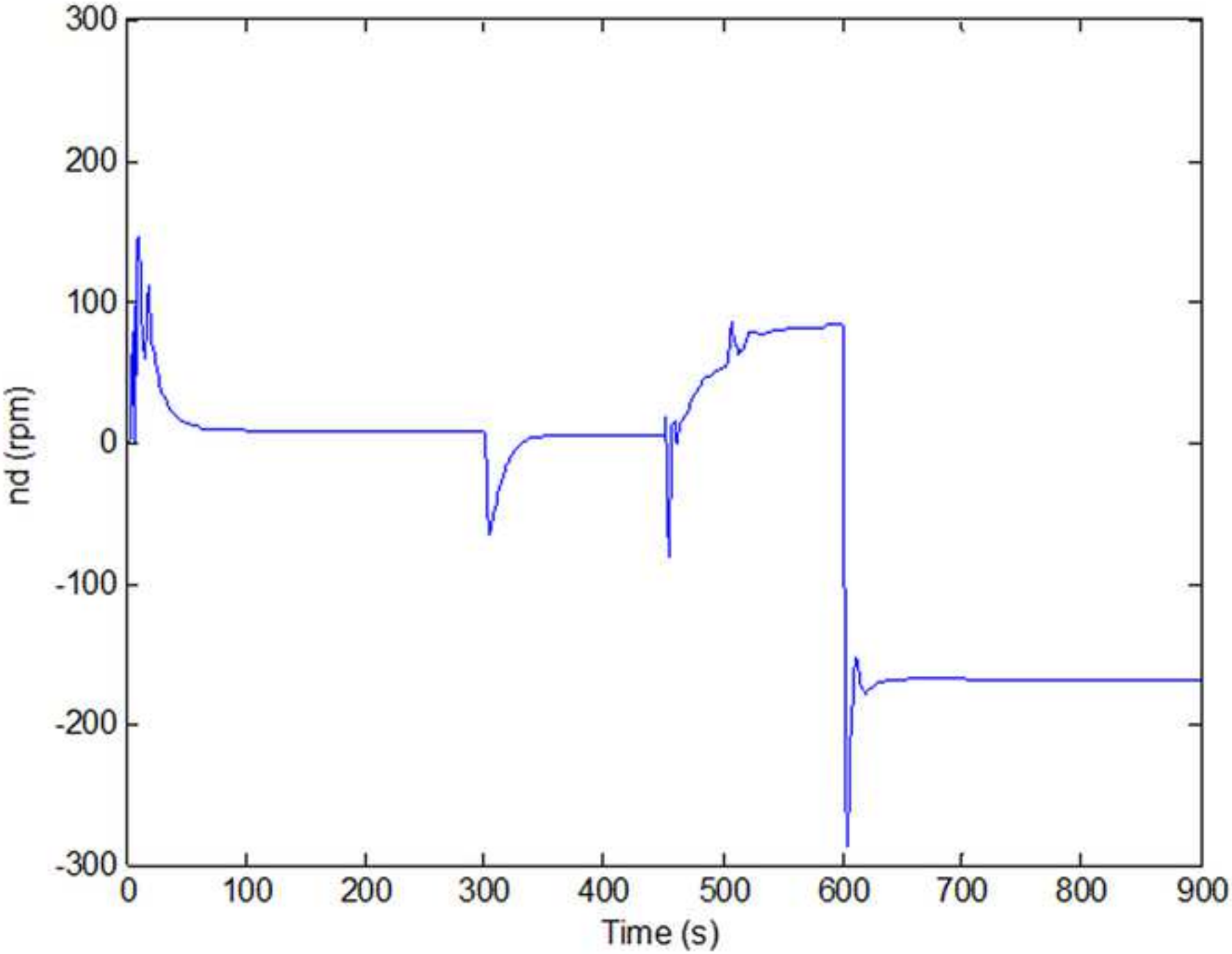
(P) re-initialised every 100 s Plant output

Click here to download high resolution image



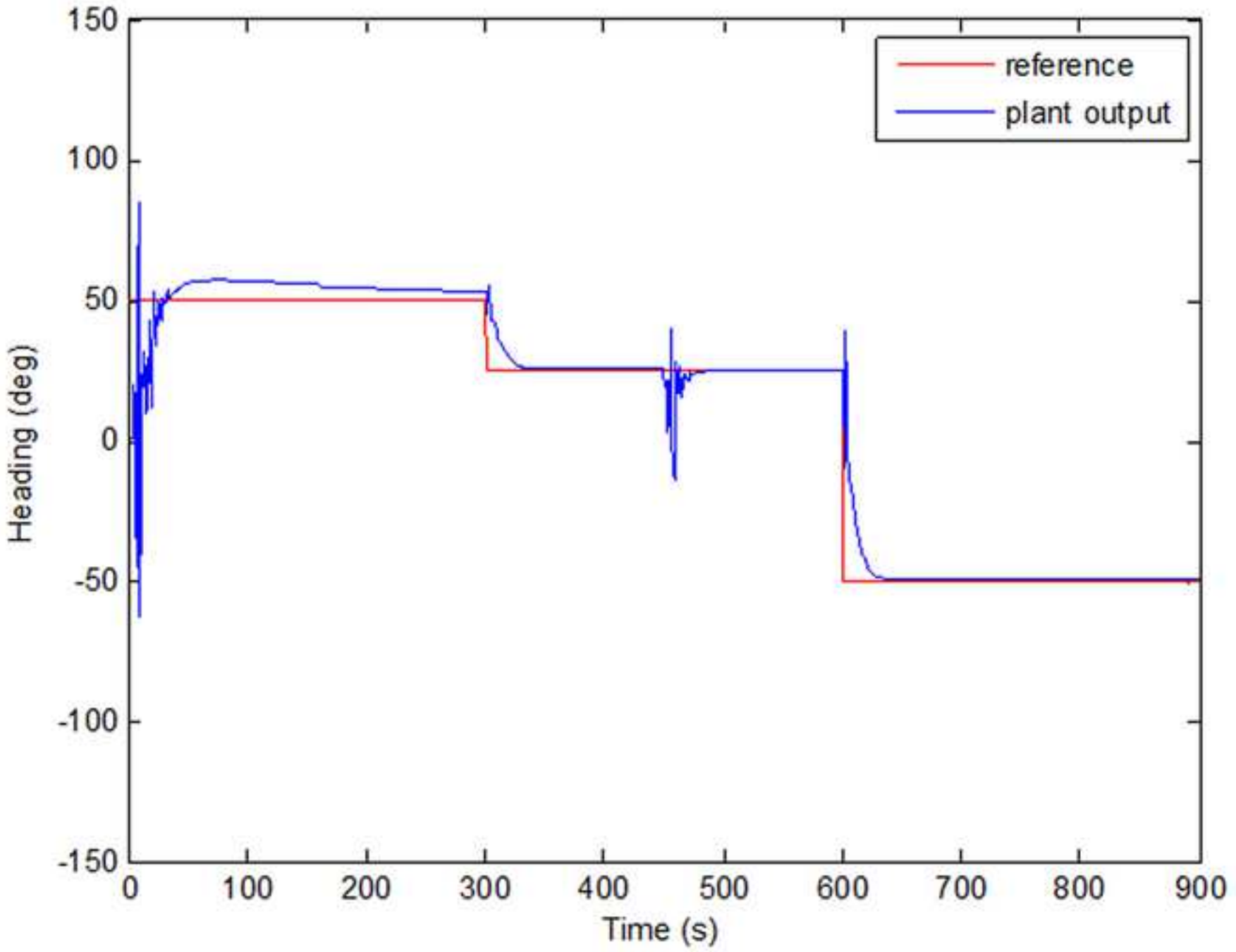
(P) re-initialised every 100 s Controller action

Click here to download high resolution image



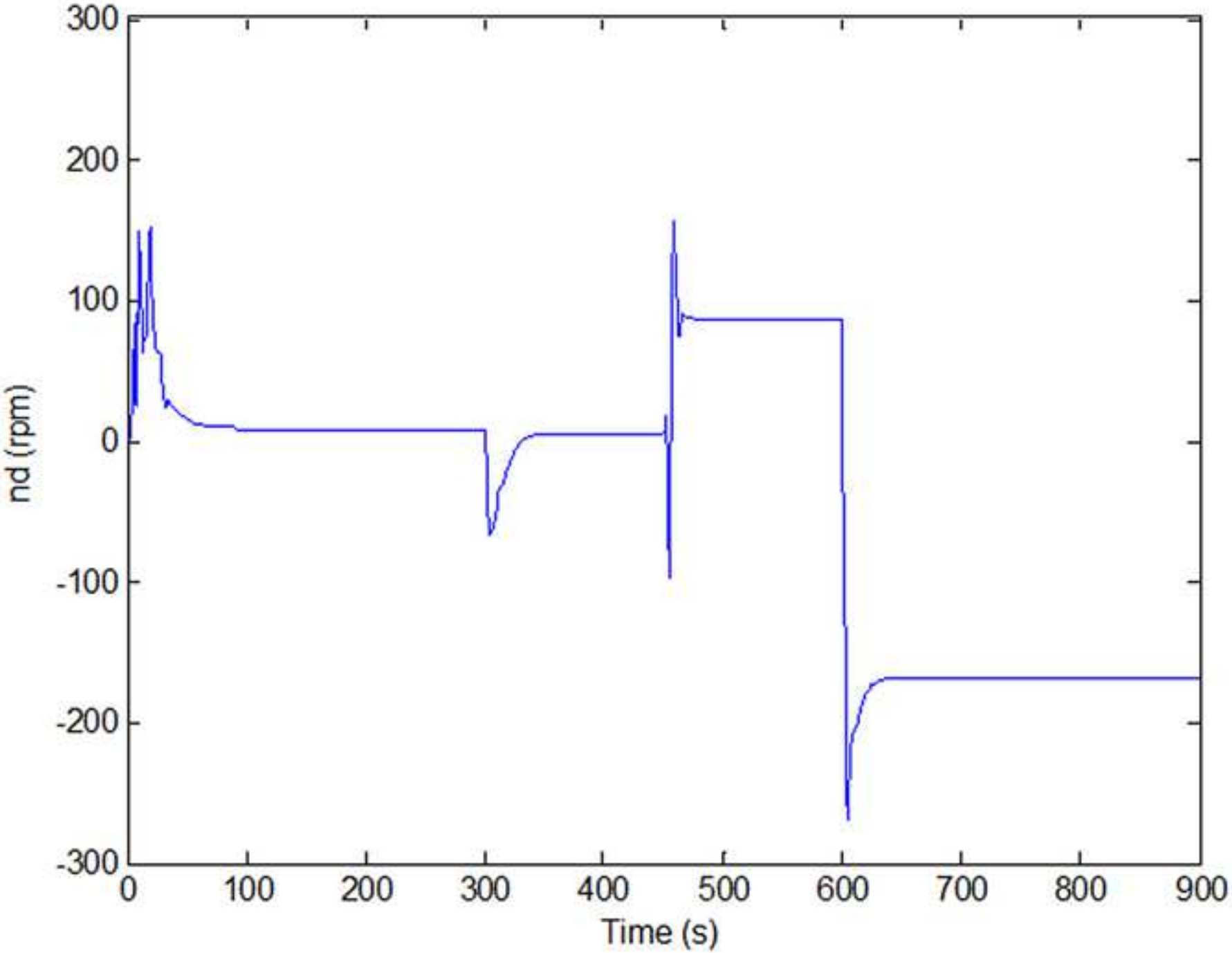
(P) re-initialised every 25 s Plant output

Click here to download high resolution image

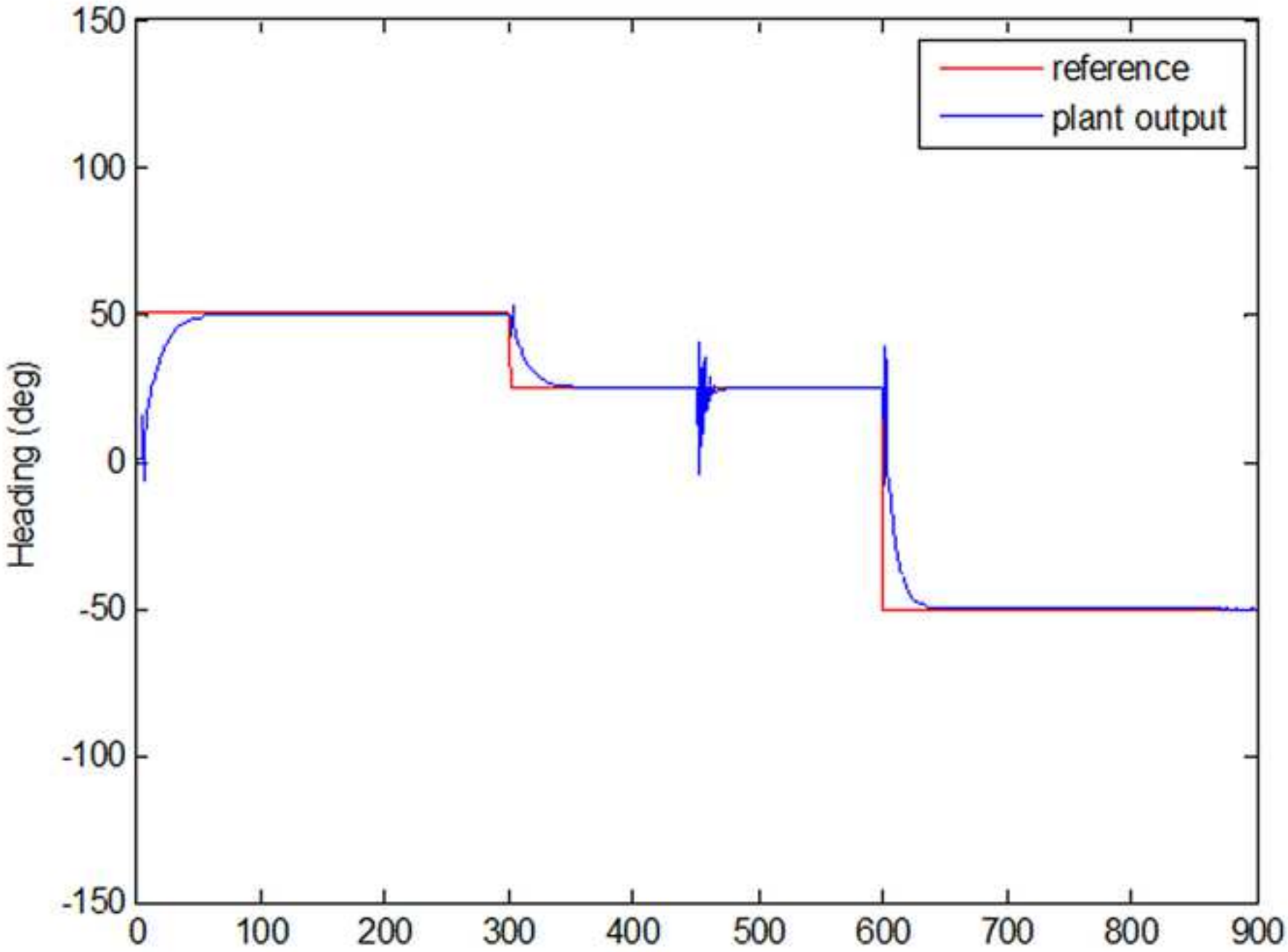


(P) re-initialised every 25 s Controller action

Click here to download high resolution image



WLS real θ plant output
Click here to download high resolution image



WLS real θ controller action

Click here to download high resolution image