

Development of a Model Predictive Controller for an Unstable Heavy Self-balancing Robot

Michał Okulski and Maciej Ławryńczuk

Institute of Control and Computation Engineering, Warsaw University of Technology

ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Email: michal@okulski.pl, M.Lawrynczuk@ia.pw.edu.pl

Abstract—This paper describes development of a control system for a heavy self-balancing two-wheeled robot. The development process includes: model identification, model tuning, design and tuning of a Model Predictive Control (MPC) algorithm. Although a simple linear state-space model with only two state variables is used, the results of laboratory experiments clearly indicate that the MPC algorithm based on such a model works well, i.e. the algorithm is able to effectively stabilise the robot.

I. INTRODUCTION

Two-wheeled self-balancing robot dynamics model contains many non-linear state equations like those described in [3], [7]. It can be also considered as a variant of the well-known inverted pendulum process [2], [6]. In the literature many approaches have been reported to control and stabilize such a robot: the classic PID controller [10], a fuzzy controller [15], the LQR controller [11] or a neural-network (non-linear) controller [12]. In addition to that a Model Predictive Control (MPC) algorithm [4], [5], [8], [13] has been also considered in [1] but effectiveness of that approach has been demonstrated in MATLAB simulations only.

In this work a laboratory heavy self-balancing two-wheeled robot at first described in [9] is considered. The objective is to design, tune and validate an MPC algorithm to effectively stabilise the robot. In order to design such a controller at first a linear state-space dynamic model of the process is developed. Model development consists of two steps: model identification and model tuning. Next, the parameters of the MPC controller are tuned. In spite of the fact that the MPC algorithm uses for process prediction a simple linear state-space model with only two state variables, the results of laboratory experiments clearly indicate that the MPC algorithm based on such a model works well, i.e. the algorithm is able to effectively stabilise the robot effectively.

II. LABORATORY SELF-BALANCING ROBOT

Fig. 1 shows a heavy, two-wheeled self-balancing robot, similar to Segway products. It has been built from scratch for research purposes. It weights more than 30kg, mostly because of its batteries (two Pb batteries) which power two DC electric motors (200W each). Motors are integrated with a 1:12 gearbox. Custom-made electronic boards take care of distributing power to the motors and the mainboard contains the Inertial Measurement Unit (IMU) with 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer. Custom sensor

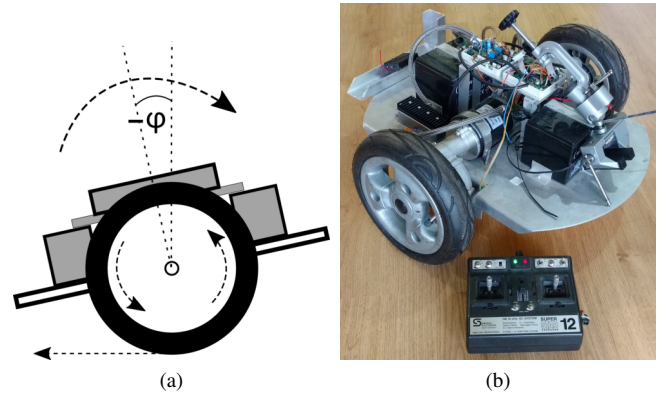


Fig. 1: The self-balancing robot: a) a schematic, b) the real robot described in this paper

filtering and fusion algorithm has been implemented [9] in the IMU due to limited computational power of its microcontroller (8-bit, Atmel's AT90CAN128, running at 14.7456 MHz). Additionally, a contactless magnetic encoder is mounted on the motor shaft directly. The mainboard sends the telemetry data at runtime via radio transceiver to a PC where the data is saved for further analysis.

Complete robot control system is a cascade of two controllers: low level (fast loop) tilt angle stabilization controller and high level (slower loop) speed controller. The first of them uses the IMU readings to keep robot tilt angle tight to the set value. The desired tilt angle is adjusted by the second controller based on motor encoder readings [9]. The high level controller is a PD structure. As the second one a PD (or PID) controller may be used as described in [9] but in this work a more advanced MPC approach is used.

III. MODEL IDENTIFICATION

A. Closed-loop Process Excitation with Switching Controller

Due to unstable nature of the robot, a classical open-loop step-response model identification experiment is not possible. In order to derive a model which would be next used in an MPC algorithm, the following close-loop approach has been taken. A simple switching controller (with hysteresis) described by the equations

$$\varphi_c(k) = \begin{cases} 8^\circ, & \text{if } \text{sgn}(\omega(k)) = \text{sgn}(\varphi(k)) \\ 5^\circ, & \text{if } \text{sgn}(\omega(k)) \neq \text{sgn}(\varphi(k)) \end{cases} \quad (1)$$

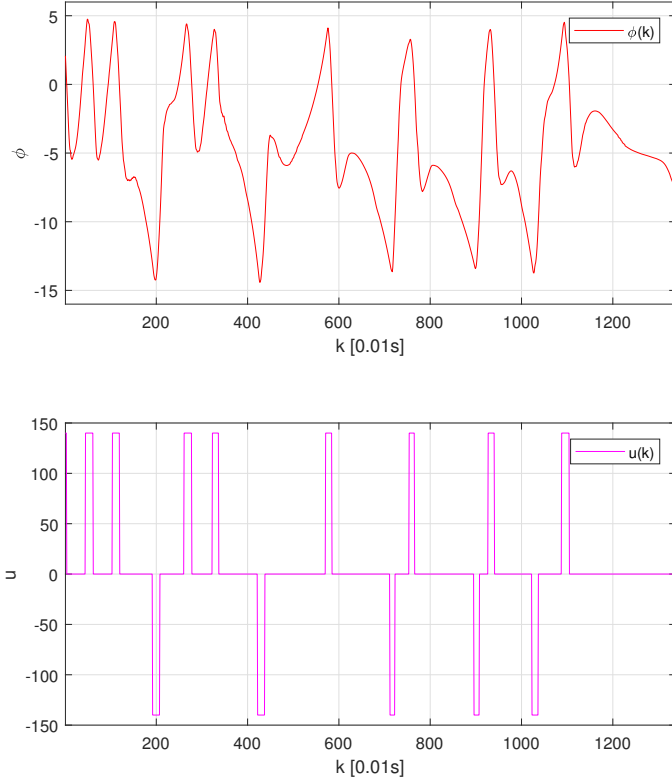


Fig. 2: Closed-loop run experiment (switching with hysteresis controller is installed): real data for model structure identification

and

$$u(k) = \begin{cases} +140, & \text{if } \varphi(k) \geq \varphi_c(k) \\ -140, & \text{if } \varphi(k) \leq -\varphi_c(k) \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

has been implemented instead. The controller has the objective to excite the process in such a way that data set for model identification is obtained. The excitations must satisfy physical limits. The obtained results of experiments are shown in Fig. 2. The process is stabilised, but it is important to note the fact that the robot is not exactly in the vertical position, variability of signals is quite significant. It is beneficial as signals' variability would allow to find a model not only good at the vertical position, but also in some neighbourhood of the equilibrium point. There are clearly visible time periods where robot falls down and time periods where constant input signal is set to the motor driver.

Using the closed-loop data shown in Fig. 2, the following state-space model with only two state variables is found

$$\begin{cases} X(k+1) &= AX(k) + Bu(k-\tau) \\ y(k) &= CX(k) \end{cases} \quad (3)$$

where model parameters are

$$A = \begin{bmatrix} 0.97 & 0.003 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -0.0007571429 \\ 0 \end{bmatrix}, \quad (4)$$

$$C = [0 \quad 1], \quad X = \begin{bmatrix} \omega \\ \varphi \end{bmatrix}$$

Model parameters have been determined using prediction error minimisation. Model input delay $\tau = 4$ has been inferred from the experiment data. Two model states are considered: the angular velocity ω and the tilt angle φ which is also the model output value. The model is simplified: e.g. non-linearities like $\sin(\varphi)$ [3] are replaced by just φ , equilibrium point hysteresis (due to soft tires with good grip) and loosely coupled motor gearboxes are neglected. As expected, the resulting model is unstable. It is natural since the real robot driven in an open-loop is unstable too.

B. Closed-loop Process Excitation with PD Controller

Another experiment has been performed to verify model accuracy: the robot has been driven in a closed-loop with a manually tuned PD Controller

$$\begin{cases} e(t) &= \varphi(t) - \varphi^{\text{set}} \\ u(t) &= K_P e(t) + K_D \frac{de(t)}{dt} \end{cases} \quad (5)$$

The coefficients K_P and K_D have been found manually. The controller has been just roughly tuned to keep robot tilt angle oscillations. That experiment gives more (non-trivial) data to validate identified models. Experimental results are presented in Fig. 3. Two sets of data have been used to validate the model, i.e. the data obtained when the PD controller is installed (Fig. 3) and when the switching with hysteresis controller is installed (Fig. 4). Fig. 4 is just a representative subset of data from the longer experiment (Fig. 2).

The model is run in the Output Error (OE) simulation model. Fig. 5 shows that due to unstable nature of the model, output error rises dramatically when the model is simulated using very long time horizon, i.e. no measurements from the real process. Fig. 6 and 7 show model simulation with much shorter horizon (just 30 time steps) from a few initial conditions. Simulation gives better results, but the model still needs to be tuned.

IV. MODEL TUNING

In order to assess model accuracy the following fitness function is defined

$$J_m = \sum_{h=0}^{H-1} \hat{Y}(h, 30) \quad (6)$$

where

$$\hat{Y}(h, s) = \sum_{k=0}^{s-1} (y(hs+k) - \hat{y}(hs+k))^2 \quad (7)$$

The objective of model tuning is to tune model's parameters in such a way that the model error is minimised. The square errors of subsequent (and independent) model simulations $\hat{Y}$

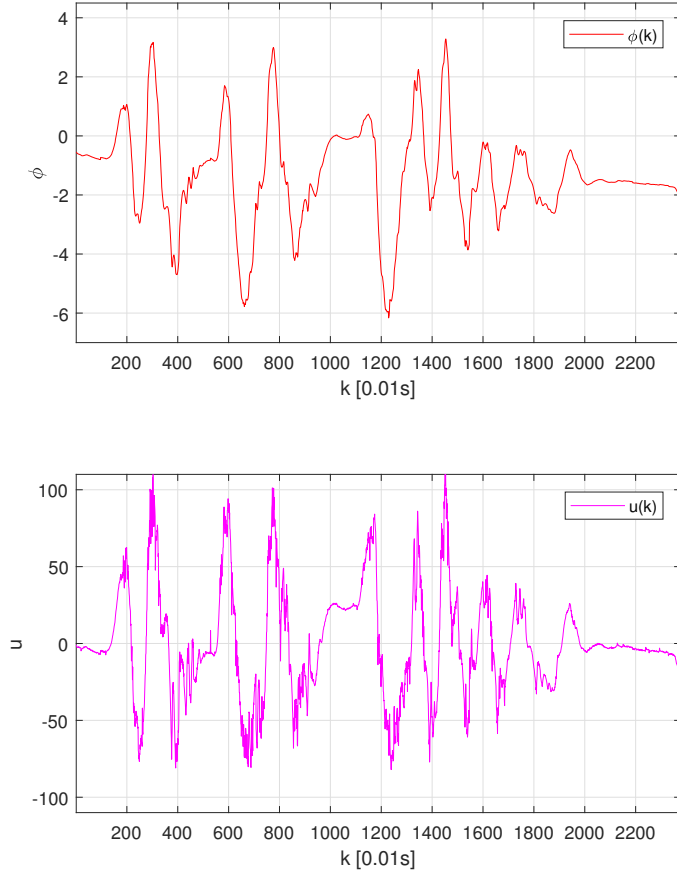


Fig. 3: Closed-loop run experiment (PD controller is installed): real data for model testing

within fixed (short) time intervals, where h is the interval start point, s is the interval length in time steps (here: 30 time steps) and H is total amount of time intervals to be considered. The fitness function defined by Eqs. (6) and (7) has been minimised using a nonlinear optimisation procedure (MATLAB's *fmincon* function has been used for this purpose). Model simulation results for data coming from both previous experiments, using short simulation horizon, are presented in Figs. 9 and 10. It is important to stress the fact that the tuned model performs very well in the OE mode on the full time window. It gives better (stable) results as demonstrated in Fig. 8, in contrast to initial (not tuned) model whose behaviour is depicted in Fig. 5.

The obtained model may be used to calculate the open-loop step-response, which may be next used in MPC. It is depicted in Fig. 11. Obviously, since the process and its state-space model (3) are unstable, a step of the manipulated variable in open-loop leads to an unstable process output trajectory.

V. MODEL PREDICTIVE CONTROLLER

A. Preliminaries

Taking into account the structure of the state-space model defined by Eqs. (3), it is necessary to develop an MPC algorithm for the process with one manipulated variable (input

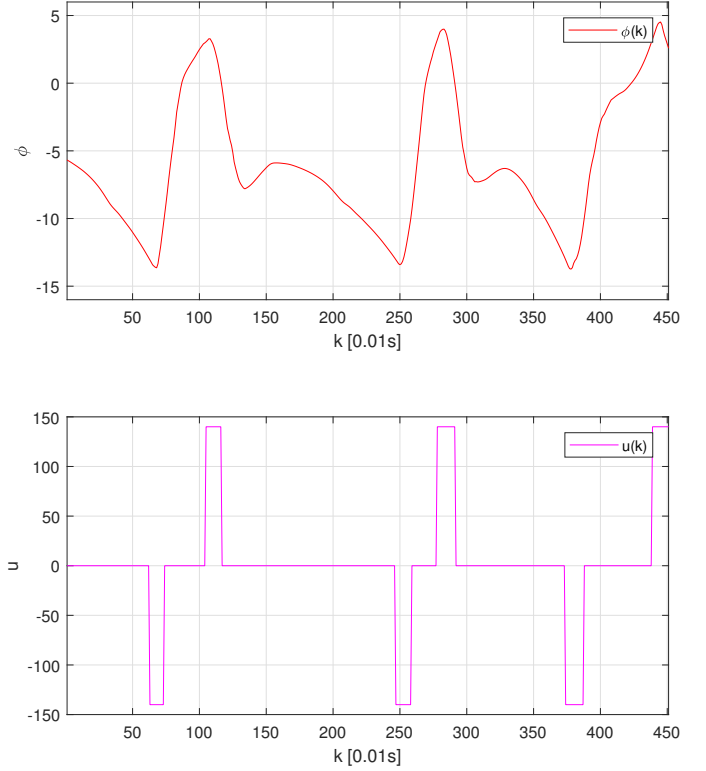


Fig. 4: Closed-loop run experiment (switching with hysteresis controller is installed): a subset of the full experiment data used for model testing

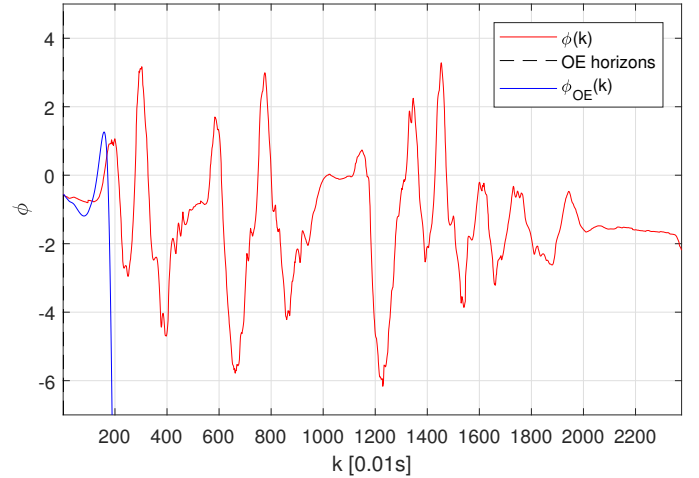


Fig. 5: Model (4) run in OE mode on real data (PD controller is installed), simulation horizon set to full time window – model is unstable

of the process), u , and one controlled variable (output of the process, y). The manipulated variable is motors power and the controlled variable is the tilt angle φ . At each discrete sampling instant, $k = 0, 1, 2, \dots$ the MPC algorithm calculates on-line a set of N_u future control increments

$$\Delta \mathbf{u}(k) = [\Delta u(k|k) \dots \Delta u(k + N_u - 1|k)]^T \quad (8)$$

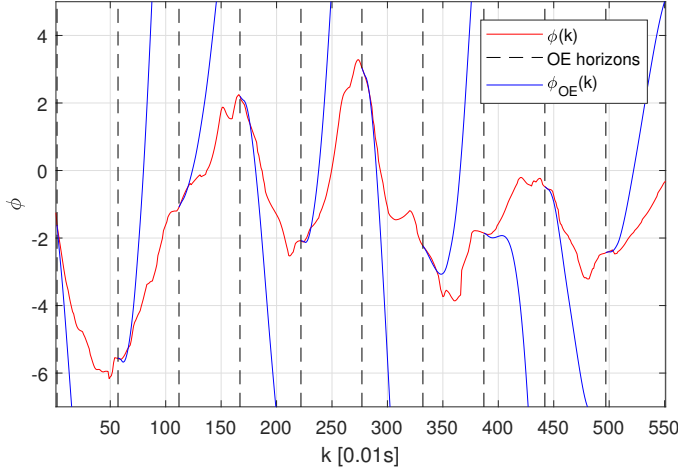


Fig. 6: Model (4) run in OE mode on real data (PD controller is installed), simulation horizon set to 30

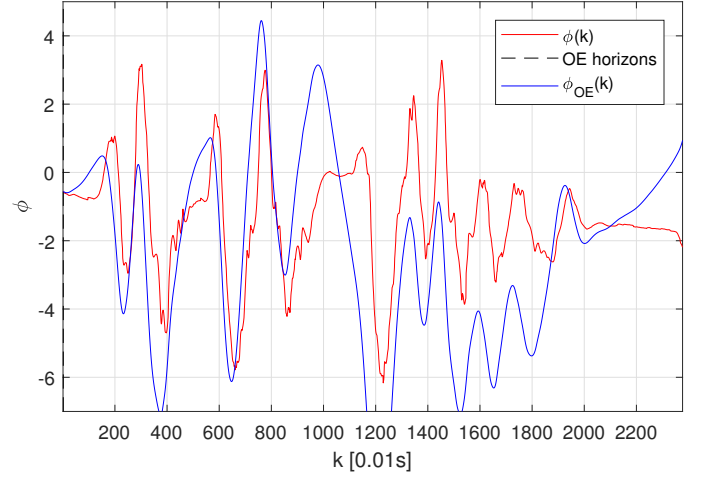


Fig. 8: Tuned model run in OE mode on real data, simulation horizon set to full time window – model is much more stable

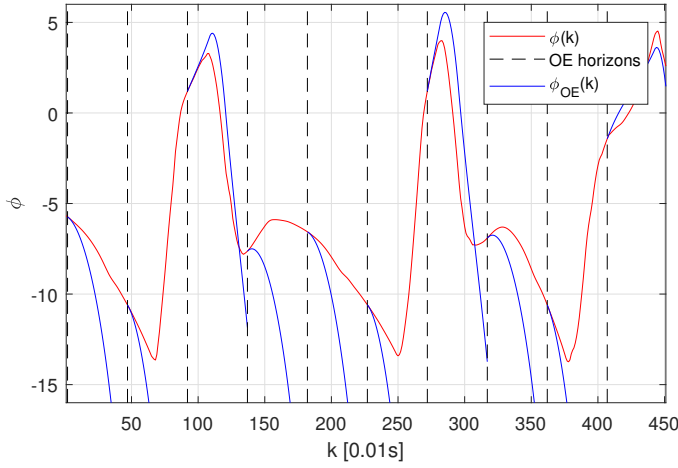


Fig. 7: Model (4) run in OE mode on real data (switching with hysteresis controller is installed), simulation horizon set to 30

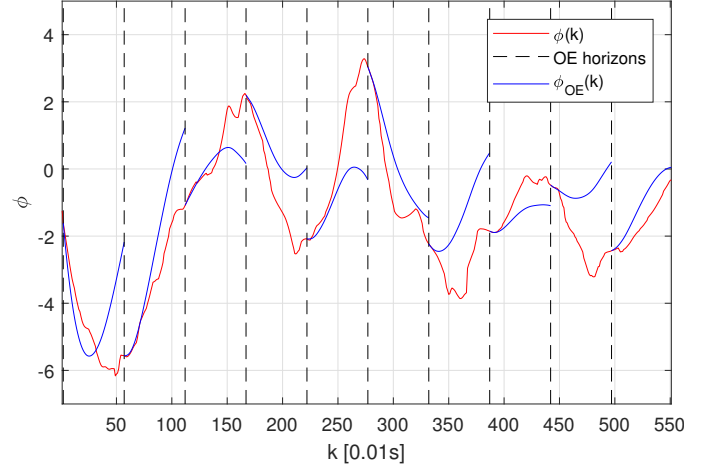


Fig. 9: Tuned model run in OE mode on real data (PD experiment), simulation horizon set to 30

where N_u is called a control horizon. The increments are defined as

$$\Delta u(k|k) = u(k|k) - u(k-1) \quad (9)$$

and

$$\Delta u(k+p|k) = u(k+p|k) - u(k+p-1|k) \quad (10)$$

for $p = 1, \dots, N_u - 1$. It is assumed that $\Delta u(k+p|k) = 0$ for $p \geq N_u$. The future increments of the manipulated variables (8) are calculated in such a way that the differences between the set-point trajectory $y^{sp}(k+p|k)$ and the predicted output values $\hat{y}(k+p|k)$ are minimised over the prediction horizon $N \geq N_u$, i.e. for $p = 1, \dots, N$. Hence, for optimisation of the future control policy (8) the following quadratic cost function

is usually used

$$J(k) = \sum_{p=1}^N \|y^{sp}(k+p|k) - \hat{y}(k+p|k)\|^2 + \lambda \sum_{p=0}^{N_u-1} \|\Delta u(k+p|k)\|^2 \quad (11)$$

where $\lambda > 0$ is a tuning coefficient (the bigger its value, the slower the set-point tracking, but it makes it possible to penalise the excessive increments of the manipulated variable). In spite of the fact that as many as N_u increments are calculated, only its first element of is actually applied to the process, i.e.

$$u(k) = \Delta u(k|k) + u(k-1) \quad (12)$$

At the next sampling instant, $k+1$, the measurement of the process output is updated, the prediction is shifted one step forward and the whole calculation procedure is repeated.

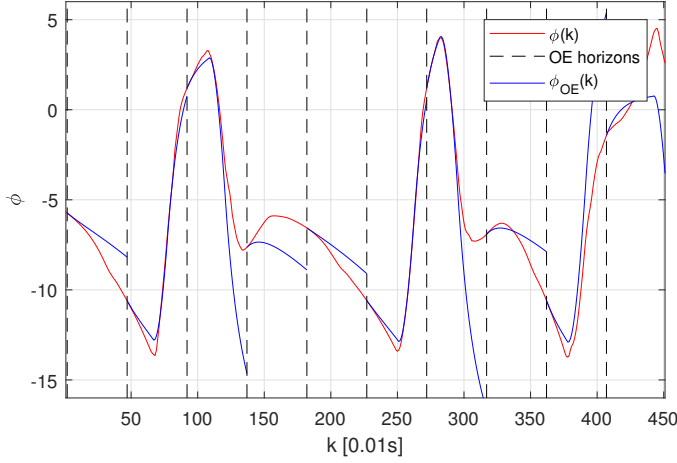


Fig. 10: Tuned model run in OE mode on real data (Switching with hysteresis experiment), simulation horizon set to 30

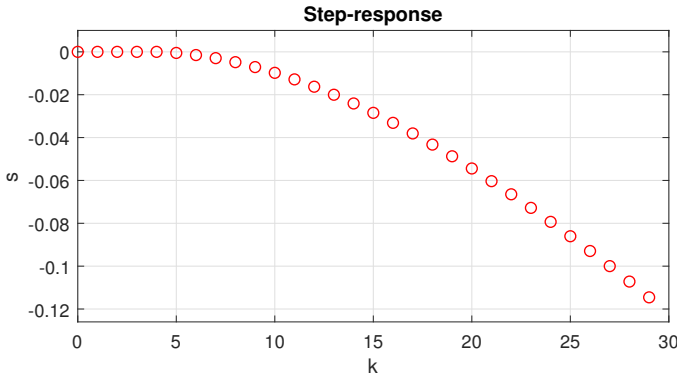


Fig. 11: Step-response of the tuned model

B. On-Line Optimisation of Control Policy

The MPC cost-function (11) may be expressed in the compact vector-matrix-notation

$$J(k) = \|\mathbf{Y}^{\text{sp}}(k) - \hat{\mathbf{Y}}(k)\|^2 + \|\Delta \mathbf{U}(k)\|_{\Lambda}^2 \quad (13)$$

where the set-point trajectory and the predicted trajectory are vectors of length N

$$\mathbf{Y}^{\text{sp}}(k) = [y(k+1|k) \dots y(k+N|k)]^T \quad (14)$$

$$\hat{\mathbf{Y}}(k) = [\hat{y}(k+1|k) \dots \hat{y}(k+N|k)]^T \quad (15)$$

Since it is assumed that the future values of the set-point trajectory over the prediction horizon are not known, it is straightforward to use only a current set-point $y^{\text{sp}}(k)$, which is constant over the prediction horizon, i.e.

$$y^{\text{sp}}(k) = y^{\text{sp}}(k+1|k) = \dots = y^{\text{sp}}(k+N|k) \quad (16)$$

In all MPC algorithms based on linear models and when a quadratic cost-function is used, it is possible to derive the vector of optimal increments of the manipulated variable in the following way [13]

$$\Delta \mathbf{U}(k) = \mathbf{K}(\mathbf{Y}^{\text{sp}}(k) - \mathbf{Y}^0(k)) \quad (17)$$

where the matrix of dimensionality $N_u \times N$ is

$$\mathbf{K} = (\mathbf{G}^T \mathbf{G} + \Lambda)^{-1} \mathbf{G}^T \quad (18)$$

The step-response matrix

$$\mathbf{G} = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ s_2 & s_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N & s_{N-1} & \dots & s_{N-N_u+1} \end{bmatrix} \quad (19)$$

is of dimensionality $N \times N_u$. The model's reaction to the past signals (up to the sampling instant k) is described by the free trajectory of the model

$$\mathbf{Y}^0(k) = [y^0(k+1|k) \dots y^0(k+N|k)]^T \quad (20)$$

The free trajectory is calculated from the state-space model (3) assuming that the manipulated variable is constant, equal to the last value applied to the process, i.e. $u(k-1)$, [13].

C. Tuning of MPC

In fact prediction horizon N and control horizon N_u can be treated as tunable parameters next to the λ coefficient. Eq. (17) shows that it is necessary to keep both horizon values as small as possible to speed-up the control law computations. It is especially important when the algorithm is implemented in an embedded system and requires a controller with high frequency of intervention (with a very short sampling period). Used MPC tuning algorithm consists of the following steps:

- 1) Start with $N = 100$, $N_u = N$, $\lambda = 1$.
- 2) Decrease λ as long as there are no oscillations of the robot tilt angle with rising amplitude.
- 3) Decrease N as long as robot behaves stable.
- 4) Decrease N_u as long as robot behaves stable.

The final parameters parameters of the MPC algorithm are: $N = 30$, $N_u = 25$, $\lambda = 0.055$.

D. Experiments

Fig. 12 shows how robot comes back to a stable state after strong manual kick-off. It takes some 2-4 seconds for the MPC controller to fully stabilise the process. It is worth mentioning that the trajectory of the manipulated variable calculated by the MPC controller is much smoother than in the case of the PD one, as shown in Fig 3. On the basis of the experiments carried out it is possible to confirm that that robot tilt angle can be successfully stabilised using the MPC algorithm described above.

VI. CONCLUSION

Design and implementation of a fast control system for a real unstable laboratory robot with complicated (unknown) dynamics is not a trivial task. This work presents simple but successful approach to solve the robot stabilisation task based on an MPC controller. At first a linear state-space model is obtained. For this purpose data must be generated in closed-loop. For data generation the robot is controlled by a simple switching controller and a PD structure. Next, an MPC

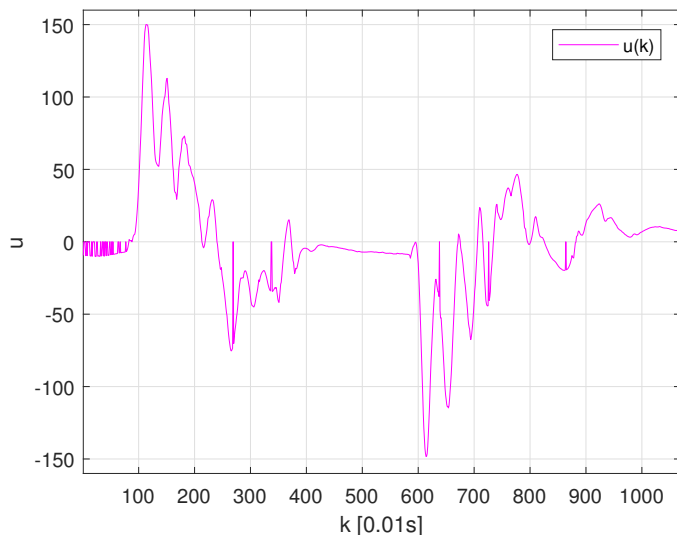
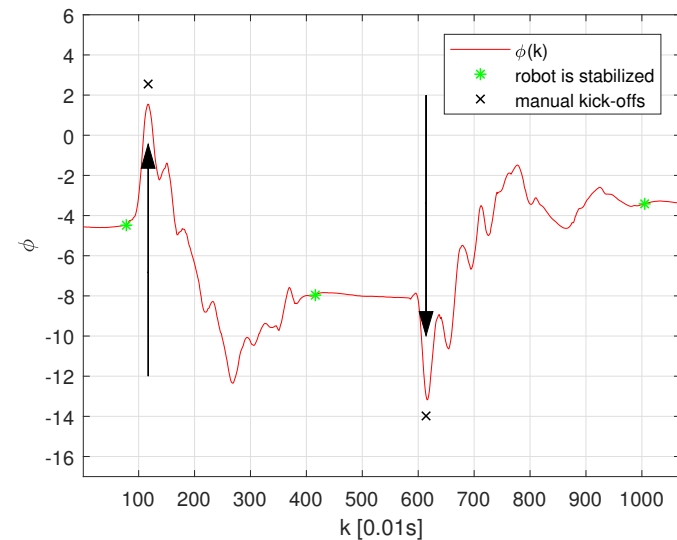


Fig. 12: Tuned MPC experiment results: manual kick-offs and stabilisation

algorithm is developed and its parameters are tuned. As it is shown, the resulting MPC controller works very effectively: it stabilises the process fast after a strong disturbance. It is worth stressing that the MPC controller works well in spite of the fact that the linear model used for prediction is only a very rough approximation of the nonlinear process.

Additional observations have been made while tuning the model and the MPC controller:

- 1) Some model identified using the available data, which gives worse prediction accuracy defined by the fitness function J_m (Eq. (6)), may be used to obtain faster MPC trajectories. It is because such models have slightly

smaller gain than that of the process and the MPC controller generates more aggressive control action.

- 2) Sensor fusion and filtering algorithm implemented in robot's Inertial Measurement Unit (IMU) can be a plausible explanation of scaled-down high frequency tilt angle changes.
- 3) Soft tires with good grip add extra hysteresis to the robot equilibrium point – it can be seen in Fig. 12. The robot stays stable at different tilt angles within range $-4^\circ \dots -8^\circ$. This effect has been not taken into account by the simplified model (Eq. (3) and (4)) but still the developed MPC algorithm is able to stabilize the robot very effectively.

Future research can focus on reducing the gap between the inferred model and the real robot by replacing the IMU unit. That should simplify precise model tuning and make the MPC even more aggressive.

REFERENCES

- [1] M. M. Azimi, H. R. Koofgar, Model predictive control for a two wheeled self balancing robot. First RSI/ISM International Conference on Robotics and Mechatronics, pp. 152-157, 2013.
- [2] K. J. Åström, R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, 2007.
- [3] H. Bin, L. W. Zhen, L. H. Feng, The Kinematics Model of a Two-Wheeled Self-Balancing Autonomous Mobile Robot and Its Simulation. Second International Conference on Computer Engineering and Applications, Vol.2, pp.64-68, 2010.
- [4] E. F. Camacho, C. Bordons, Model Predictive Control. Springer, London, 1999.
- [5] C.R. Cutler, B.L. Ramaker, Dynamic matrix control—a computer control algorithm. Proceedings of the AIChE National Meeting, Houston, 1979.
- [6] G. F. Franklin, J. D. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems. Pearson, 2015.
- [7] S. Haddout, Nonlinear reduced dynamics modelling and simulation of two-wheeled self-balancing mobile robot: SEGWAY system. Systems Science & Control Engineering, p.1-11, 2017.
- [8] J. M. Maciejowski, Predictive Control with Constraints. Prentice Hall, Harlow, 2002.
- [9] M. Okulski, M. Ławryńczuk, A cascade PD controller for heavy self-balancing robot, Automation 2018, Advances in Intelligent Systems and Computing, vol. 743, ed.: Szewczyk R., Zieliński C., Kaliczyńska M., pp. 183–192, Springer, Heidelberg, 2018.
- [10] M. Majczak, P. Wawrzyński: Comparison of two efficient control strategies for two-wheeled balancing robot. The 20th International Conference on Methods & Models in Automation & Robotics (MMAR), Międzyzdroje, Poland, pp. 745-746, 2015.
- [11] F. Sun, Z. Yu, H. Yang: A design for two-wheeled self-balancing robot based on Kalman filter and LQR. International Conference on Mechatronics and Control (ICMC), pp. 612-616, 2014.
- [12] C. Sun, T. Lu, K. Yuan: Balance control of two-wheeled self-balancing robot based on Linear Quadratic Regulator and Neural Network. Fourth International Conference on Intelligent Control and Information Processing (ICICIP), pp. 862-867, 2013.
- [13] P. Tatjewski, Advanced Control of Industrial Processes, Structures and Algorithms. Springer, London, 2007.
- [14] H. T. Teixeira, V. Semedo de Mattos Siqueira, C. J. Munaro: Comparison of methods for estimation and compensation of friction applied to an inverted pendulum., 9th IEEE International Conference on Control and Automation (ICCA), pp. 818-822, 2011.
- [15] Q. Yong, L. Yanlong, Z. Xizhe, L. Ji: Balance control of two-wheeled self-balancing mobile robot based on TS fuzzy model. International Forum on Strategic Research, pp. 406-409, 2011.