BALANCING WHEELED ROBOT: EFFECTIVE MODELLING, SENSORY PROCESSING AND SIMPLIFIED CONTROL

Robert Grepl*

This paper deals with the modelling and control of balanced wheeled autonomous mobile robot. For the MBS dynamics modelling software tool Matlab-SimMechanics is used. The model derived automatically from geometric-topological description of MBS is used for the control purposes (local linearization for state space control, testing of nonlinear system controlled by LQR) and also as a reference during the analytical model formulation for global feedback linearization. The dual accelerometer is used as a tilt sensor and the proposed method for sensory processing is described in this paper. The approach is based on iterative solution of nonlinear equation. Control using the state space (LQR) and the feedback linearization is compared. Also, the influence of sensor noises and delays implemented into the model are discussed. Finaly, the solution is verified on real physical model controlled by means of hardware in the loop (HIL).

Keywords: *simulation, dynamics, Matlab-Simulink, SimMechanics, inverted pendulum, mobile robot, state space control, feedback linearization*

1. Introduction

Modelling and control of balancing wheeled autonomous mobile robots recently have been attracting the attention of researches. The Segway Inc. successfully brings the ideas to the commercial implementation. Several projects deal with the utilization of human transport device as an autonomous robot [10].

The aims of work described in this contribution can be stated as follows: a) use as minimal number of sensor as possible; b) apply maximally fast design process (use SimMechanics for dynamic modelling); c) design a controller based on feedback linearization paradigm; d) add the necessary noise and delays into the simulating model and compare the proposed controller to the LQR one; and e) verify the simulation results on the physical laboratory model of balanced robot.

The modelling process of multi body system (MBS) dynamics is composed of three main phases: 1) simplification of real system to geometric-topological description of rigid MBS (understood as graph); 2) derivation of mathematical model (dynamic formalisms lead to ODE (ordinary differential equations) or DAE (differential–algebraic equations)); 3) solution of motion equations.

The first step is naturally esential, all negligible properties of real system (body deformation, real friction aspects) are excluded from modelling. The last third step is usually completed by routine numerical integration on the computer (if one has usable equations).

* Ing. R. Grepl, Ph.D., Institute of Solid Mechanics, Mechatronics and Biomechanics, Faculty of Mechanical Engineering, Brno University of Technology, Czech Republic

The second phase normally means the 'hand' application of Lagrange or Newton approach and the system of ODE or DAE is obtained. The probability of a mistake grows rather quickly with the complexity of modelled system. SimMechanics is one of the useful software tools for automatical equation formulation [5]. Its main advantage is a direct integration into the Simulink environment. In this paper, we would like to highlight the importance and usefullness of such tools for practical engineering application.

The plant shown in Fig. 1 can be modelled as two rigid bodies connected by means of rotational joint.

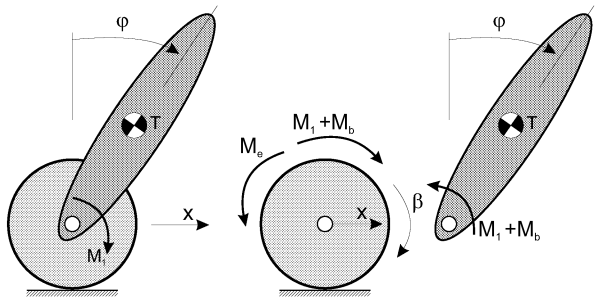

Fig.1: The system schema and torques acting on both bodies

Similarly as in [2], [13], [14], [16] and others, we consider an ideal rolling contact between the wheel and surface. First we tested the state space control as in [13]. Next we implemented the feedback linearization concept [1], [12], [7]. After that, the simulation model was improved by sensory noises and computational delays.

Most of researchers use the gyroscope as the pendulum tilt sensor [2]; despite of it we implemented and tested the computational algorithm for online sensory processing of accelerometer data.

2. Fast design of dynamic model in SimMechanics

2.1. SimMechanics

SimMechanics is a relatively new extension of the standard simulating tool Matlab/Simulink. The software is intended for the building and simulation of statics, kinematics and dynamics of MBS. The main contribution for the user is an automatic building of the mathematical model; the user defines the geometry and topology of the system only. The interaction between the mechanical model and other physical domain is natural through the use of 'sensors' and 'actuators'.

Internally, SimMechanics uses relative coordinates. This leads to the tree structure of MBS (open chain/tree topology) and the related dynamic formalism. Close loop systems are solved by cutting loops and introducing additional constraints. The difficulties linked to such an approach are well known [6] but designers of SimMechanics were limited by the parent Simulink capabilities – the resulting mathematical description of MBS must be an ODE system [5]. Therefore, there was no possibility to implement the modern approach with natural coordinates and the equations in a descriptor form leading to DAE.

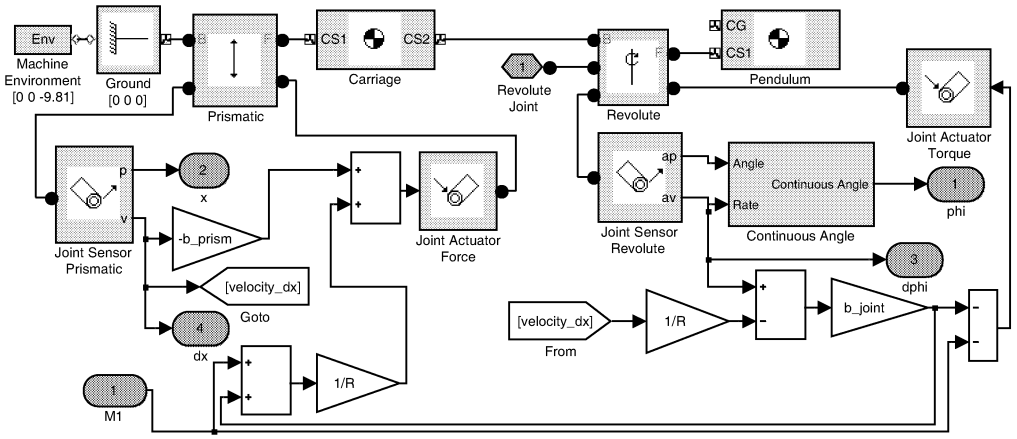

Fig.2: Simulating model of the plant built in Matlab/SimMechanics

2.2. Rolling constraint in SimMechanics

The precise and effective modelling of rolling contact between the two bodies (including potential lateral and longitudinal slip) is a rather difficult task. Also, in SimMechanics there is no straightforward solution. Relatively easy way to implement can be obtained in SimMechanics using a dynamically equivalent shift of the contact force into the centre of rotation and the corresponding compensation by a moment [15]. Thus one can model the sliping based on arbitrary advanced Coulomb models (e.g. with Stribeck effect) and depending on variable normal force.

2.3. Final model

However, in our case, the slipping is neglected and we can use a simple solution shown in Fig. 3. Rolling is reduced to translation along the x axis with the following m_{red} :

$$E_{\mathrm{k}} = \frac{1}{2}\, m\, x^2 + \frac{1}{2}\, I_{\mathrm{T}}\, \omega^2 = \frac{1}{2} \underbrace{\left(m + I_{\mathrm{T}}\, \frac{1}{R^2} \right)}_{m_{\mathrm{red}}} x^2 \ . \tag{1}$$

Let us note that

$$\vartheta = \varphi - \beta = \varphi - \frac{x}{R} \ , \tag{2}$$

$$\dot{\vartheta} = \dot{\varphi} - \frac{\dot{x}}{R} \ , \tag{3}$$

where ϑ is the angle measured on the motor shaft (using encoder), φ is the absolute inclination of pendulum and β is the angle of rolling of the wheel.

The M_{21} is the torque on the shaft while the M_{b} and M_{e} represent the viscous damping on the shaft and translational movement respectively.

$$M_{\mathrm{b}} = b \left(\dot{\varphi} - \frac{\dot{x}}{R} \right) \ , \tag{4}$$

$$M_{\mathrm{e}} = b_{\mathrm{e}}\, \dot{x} \ . \tag{5}$$

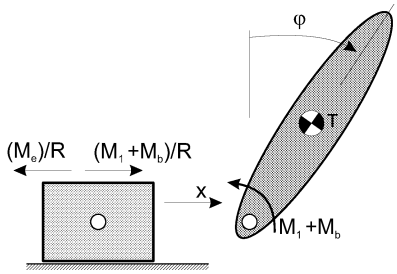

Fig.3: The model of system with rolling reduced on translation

The resulting simulation model (more details in help of SimMechanics online) is shown in Fig. 3.

3. Model of sensors

The maximal simplicity of sensory system has been given as an important requirement. The minimal reasonable configuration of sensors is the use of dual accelerometer and incremental encoder on the motor shaft. Note that the angle measured by encoder is given by eq. 2. The values measured by accelerometer depend on its position which is discussed further.

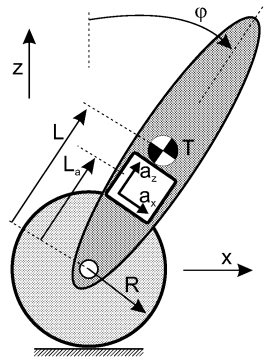

Fig.4: Dual accelerometer location

3.1. Analytical model of accelerometer

Consider the dual axis accelerometer fixed to the pendulum according to Fig. 4. The measured acceleration is in vector form

$$\vec{a} = -\vec{a}_{A1} + \vec{g}_1 \ , \tag{6}$$

$$\vec{a}_{\mathrm{A1}} = \ddot{\vec{x}} + L_{\mathrm{a}}\,\ddot{\vec{\varphi}} + L_{\mathrm{a}}\,\dot{\vec{\varphi}}^2 \ . \tag{7}$$

The acceleration in the local coordinate system of pendulum is then

$$\vec{a}_{\mathrm{A1}} = \begin{bmatrix} L_{\mathrm{a}}\,\ddot{\varphi} + \ddot{x}\,\cos\varphi \\ -L_{\mathrm{a}}\,\dot{\varphi}^2 + \ddot{x}\,\sin\varphi \end{bmatrix} \ , \tag{8}$$

$$\mathbf{g}_2 = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} 0 \\ -9.81 \end{bmatrix} = \begin{bmatrix} g\,\sin\varphi \\ -g\,\cos\varphi \end{bmatrix} \ , \tag{9}$$

where $g = +9.81$. The resulting measured acceleration in the respective axis is

$$a_{\mathrm{x}} = -L_{\mathrm{a}}\,\ddot{\varphi} - \ddot{x}\,\cos\varphi + g\,\sin\varphi\ , \tag{10}$$

$$a_{\mathrm{z}} = L_{\mathrm{a}}\,\dot{\varphi}^2 - \ddot{x}\,\sin\varphi - g\,\cos\varphi\ . \tag{11}$$

Let us consider $L_{\mathrm{a}} = 0$ (accelerometer in the axis of rotation). The final model of accelerometer is then in the form of

$$a_{\mathrm{x}} = -\ddot{x}\,\cos\varphi + g\,\sin\varphi\ , \tag{12}$$

$$a_{\mathrm{z}} = -\ddot{x}\,\sin\varphi - g\,\cos\varphi\ . \tag{13}$$

3.2. Model of accelerometer in SimMechanics

In SimMechanics, the vector $a_{\mathrm{A2}}^{\vec{}}$ can be directly measured using the block `Body Sensor`. According to 9, the vector $\vec{g}$ should be transformed into the coordinate system 2 using the rotational matrix sensed on the body.

In a practical simulating model an additional noise should be applied to the output of accelerometer.

3.3. Accelerometer used as inclinometer

The LQR as well as the feedback linearization require the measuring or reconstruction of state vector (observer design). As it is shown in Fig. 6, the output of the system including sensors is vector $\mathbf{z} = [\vartheta, a_{\mathrm{x}}, a_{\mathrm{z}}]^{\mathrm{T}}$. The following text clarifies that this output is not suitable for the design of state estimator. Therefore, the inverse task is to be solved: how to obtain the value of φ from measured a_{x} and a_{z}.

Using the matrix notation of 12 and 13 one can write

$$\begin{bmatrix} a_{\mathrm{x}} \\ a_{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} -\ddot{x} \\ -g \end{bmatrix}\ , \tag{14}$$

$$\begin{bmatrix} a_{\mathrm{x}} \\ a_{\mathrm{z}} \end{bmatrix} = \mathbf{R}_{\varphi} \begin{bmatrix} -\ddot{x} \\ -g \end{bmatrix}\ , \tag{15}$$

where $\mathbf{R}_{\varphi}$ is rotation matrix. Its important property is the orthogonality and thus

$$\mathbf{R}_{\varphi}^{-1} = \mathbf{R}_{\varphi}^{\mathrm{T}}\ , \tag{16}$$

$$\begin{bmatrix} -\ddot{x} \\ -g \end{bmatrix} = \mathbf{R}_{\varphi}^{\mathrm{T}} \begin{bmatrix} a_{\mathrm{x}} \\ a_{\mathrm{z}} \end{bmatrix}\ . \tag{17}$$

The second equation in scalar form is

$$a_{\mathrm{x}}\,\sin\varphi - a_{\mathrm{z}}\,\cos\varphi - g = 0\ . \tag{18}$$

We can solve this nonlinear algebraic equation numerically and obtain the unknown variable φ. By the numerical experiments, one easily spots two problems: 1) the equation 18 has two solutions (therefore the proper choice of initial guess is important); but mainly

2) in special circumstances there is only one solution where two trajectories intersect. Naturally, in the intersection point, the solution can incline to the wrong trajectory and so the computation of φ fails.

Let us consider the Newton's method as the most simple and computationally fastest approach (in case of need for more advanced method, the principle remains)

$$\varphi_{n+1} = \varphi_n - \frac{f}{f'} \ . \tag{19}$$

The method fails in $f' = 0$, in our case

$$f' = 0 = a_\mathrm{x} \cos\varphi + a_\mathrm{z} \sin\varphi \ , \tag{20}$$

$$\tan\varphi = -\frac{a_\mathrm{x}}{a_\mathrm{z}} \ . \tag{21}$$

An example of numerical experiment accomplished in Matlab environment using the function `fsolve` (implementation of various advanced methods) is shown in Fig. 5

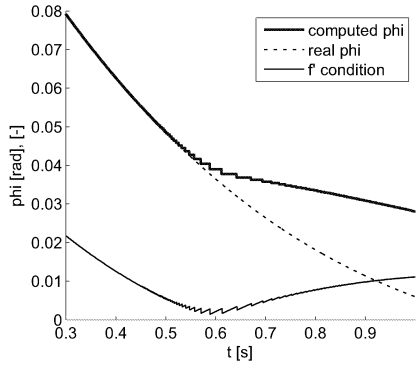

Fig.5: Example of trajectory bifurcation near the singular point (numerical solution using `fsolve` *in Matlab)*

As a practical solution of the described problem, we can recommend : 1) the initial guess used by Newton's method is calculated from the two last φ values

$$\varphi_n = \frac{\varphi_{n-1} - \varphi_{n-2}}{\Delta t}\,\Delta t + \varphi_{n-1} = 2\,\varphi_{n-1} - \varphi_{n-2} \ ; \tag{22}$$

2) as an alternative solution φ_2 near $f' = 0$ is used the simplified equation ($\ddot{x}$ neglected)

$$\varphi_n = \arcsin\,\frac{a_\mathrm{x}}{g} \ , \tag{23}$$

and finaly 3) the two above mentioned results are combined using the weight factor as follows

$$\varepsilon = \left|\tan\varphi_0 + \frac{a_\mathrm{x}}{a_\mathrm{z}}\right| \ , \tag{24}$$

$$w_2 = -\frac{1}{h}\,\varepsilon + 1 \ , \tag{25}$$

$$w_1 = 1 - w_2 \ , \tag{26}$$

$$\varphi = w_1\,\varphi_1 + w_2\,\varphi_2 \ , \tag{27}$$

where h is the range of φ_2 significance in ϵ metrics, φ_1 is the resulting angle from Newton's method.

The resulting pseudocode is rather simple:

```
while abs(f1) > 0.001
    phi = x1;
    f1 = ax*sin(phi) - az*cos(phi) - g;
    df1 = ax*cos(phi) + az*sin(phi);
    if abs(df1) < 1e-4
        x1 = asin(ax/g);
    else
        x1 = x1 - f1/df1;
    end
end
```

The critical issue is the computational time required for the described method. In our experiments, the usable value of 0.8 ms has been reached (implemented in Atmel AVR, 16 MHz, table approximation of sin).

The schema of sensory processing is shown in Fig. 6.

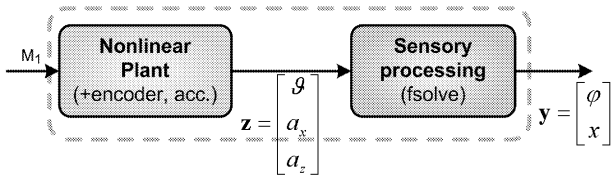

Fig.6: Schema of sensory processing

Including the derived procedure into plant model, we obtain appropriate condition to design a controller and state estimator, as the system output is $\mathbf{y} = [\varphi, x]^{\mathrm{T}}$.

4. State space control

In this section, we briefly inform the reader about the particular application of the routine procedure in the design of linear-quadratic state feedback regulator (LQR).

4.1. Linear-quadratic state feedback regulator

The LQR design works with the linear time-invariant state space model (LTI). SimMechanics allows the use of comfortable Simulink tool for automatic linearization `linmod`. Furthermore, we should identify the order of state variables in the resulting state space form using the command `StateVectorMgr.StateNames`.

After that we can easily obtain the control matrix K and weight matrix N shown in Fig. 7 (eg. [4]).

4.2. Observer design

After the sensory processing, we have only $\mathbf{y} = [\varphi, x]^{\mathrm{T}}$ therefore we need a state estimator. Although our system is nonlinear, let us use a common Luenberger observer design approach.

The schema of resulting system is shown in Fig. 7.

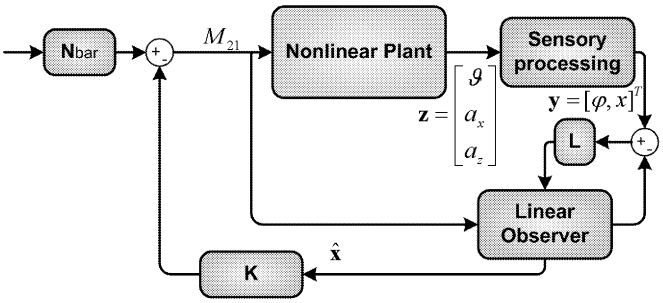

Fig.7: Schema of control of nonlinear plant using linearly designed compensator K and observer L

Next, we can verify the response of nonlinear (SimMechanics) plant and compare it to the linear (designed) one. Fig. 8 shows an example of such testing for the particular initial conditions.

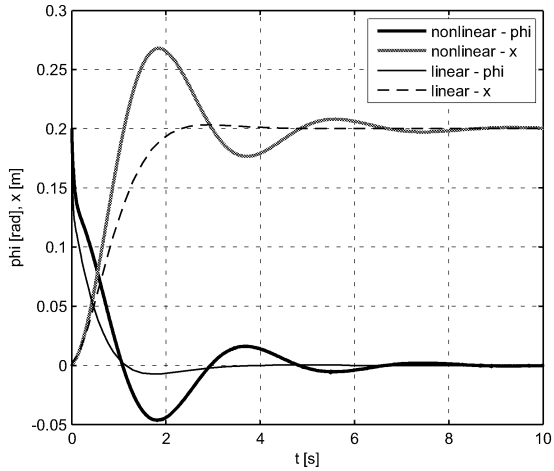

Fig.8: Comparison of the controled system response (linear and nonlinear)

In conclusion, we can state, that the nonlinearity and the sensory processing influence the behaviour of system but the LQR design still can be employed.

5. Feedback linearization

5.1. Analytical dynamic model

We choose the following generalized coordinates

$$\mathbf{q} = [\varphi, x]^{\mathrm{T}} \tag{28}$$

and derive the equation of motion using the Lagrange approach. First, we need the velocity of the center of gravity of pendulum v_2.

$$\vec{v}_2 = \vec{\dot{x}} + \vec{v}_{21} = \vec{\dot{x}} + L\vec{\dot{\varphi}} \ , \tag{29}$$

$$v_2^2 = \dot{x}^2 + 2\,L\,\dot{x}\,\dot{\varphi}\,\cos\varphi + L^2\,\dot{\varphi}^2 \ . \tag{30}$$

After simplification, the kinetic energy is

$$E_{\mathrm{k}} = \frac{1}{2}\, c_1\, \dot{x}^2 + c_2\, \cos\varphi\, \dot{x}\, \dot{\varphi} + \frac{1}{2}\, c_3\, \dot{\varphi}^2 \ , \tag{31}$$

$$c_1 = m_1 + \frac{I_{1\mathrm{A}}}{R^2} + m_2 \ , \tag{32}$$

$$c_2 = m_2\, L \ , \tag{33}$$

$$c_3 = m_2\, L^2 + I_{2\mathrm{T}} \ . \tag{34}$$

Potential energy is

$$E_{\mathrm{p}} = m_2\, g\, L\, \cos\varphi \ . \tag{35}$$

The torque M_{21} effects both the pendulum and wheels, as well as the joint viscous damping M_{b}.

$$M_{\mathrm{b}} = b\, \dot{\vartheta} = b\, \left(\dot{\varphi} - \dot{\beta}\right) = b\, \left(\dot{\varphi} - \frac{\dot{x}}{R}\right) \ . \tag{36}$$

The viscous damping of translational movement of wheels is

$$M_{\mathrm{e}} = b_e\, R\, \dot{x} \ . \tag{37}$$

Using the principle of virtual work and Lagrange equation we derive the two equations of motion

$$\mathbf{M}(\varphi)\, \ddot{\mathbf{q}} + \mathbf{B}\, \dot{\mathbf{q}} + \mathbf{f}(\varphi, \dot{\varphi}) = \mathbf{S}\, M_{21} = \mathbf{u} \ , \tag{38}$$

where

$$\mathbf{M} = \begin{bmatrix} c_3 & c_2\, \cos\varphi \\ c_2\, \cos\varphi & c_1 \end{bmatrix} \ , \tag{39}$$

$$\mathbf{B} = \begin{bmatrix} b & -\frac{b}{R} \\ -\frac{b}{R} & \frac{b}{R^2} + b_{\mathrm{e}} \end{bmatrix} \ , \tag{40}$$

$$\mathbf{f} = \begin{bmatrix} -m_2\, g\, L\, \sin\varphi \\ -c_2\, \sin\varphi\, \dot{\varphi}^2 \end{bmatrix} \ , \tag{41}$$

$$\mathbf{S} = \begin{bmatrix} -1 \\ \frac{1}{R} \end{bmatrix} \ . \tag{42}$$

Note here that even in the case of such a relatively simple system there is still a rather high potential of a mistake made by the human during derivation of motion equations. Practical and effective approach is to compare the derived model to the automatically built one by SimMechanics (or other fundamentally similar) software.

5.2. Linearization

Based on 38, we can define the input u as follows (principle of feedback linearization [1], [7], [3].

$$\mathbf{u} = \mathbf{M}(\varphi)\, \mathbf{a}_{\mathrm{q}} + \mathbf{B}\, \dot{\mathbf{q}} + \mathbf{f}(\varphi, \dot{\varphi}) \ , \tag{43}$$

where the $\mathbf{a}_{\mathrm{q}}$ is the new system input. Finally, we have a linear system of the second order

$$\ddot{\mathbf{q}} = \mathbf{a}_{\mathrm{q}} \ . \tag{44}$$

The following two issues are discussed in the next section: the controller design and the state measuring.

Normally, feedback linearization leads to n decoupled system, however, our system is underactuated (according to eq. 38 and 42). Despite the important results of underactuated systems control (e.g. [9]), our main focus is to design as simple control approach as possible. Therefore we just adopt ordinary feedback linearization paradigm and implement the control using

$$M_{21} = [-1, 0]\,\mathbf{u}\ . \tag{45}$$

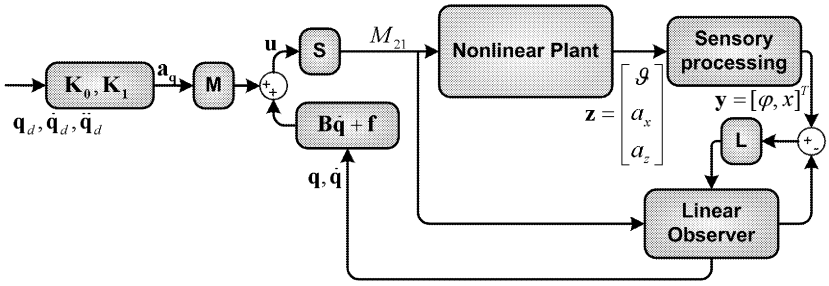

Fig.9: Schema of control using feedback linearization with linear state space observer

5.3. Controller and Observer

According to eg. [1], [3] we can define the new input vector $\mathbf{a}_\mathrm{q}$ as follows

$$\mathbf{a}_\mathrm{q} = \ddot{\mathbf{q}}^d - \mathbf{K}_0\,\tilde{\mathbf{q}} - \mathbf{K}_1\,\dot{\tilde{\mathbf{q}}}\ . \tag{46}$$

For the inverse dynamic 44, we need the full state space vector $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^\mathrm{T}$. In this paper we propose the use of Luenberger observer designed in 4.2 for our feedback linearized system. The estimated state vector $\mathbf{x}$ is then used in the controller 46. After fine tuning of parameters, we can get the cotroller that is usable, simple to design and on-line computable.

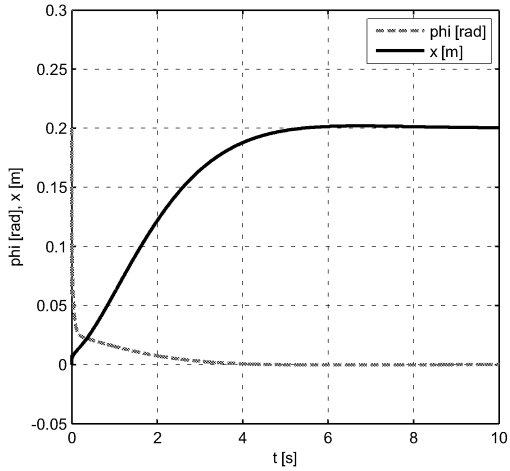

Fig.10: Response of system controlled by feedback linearization (no delays and no noise modelled)

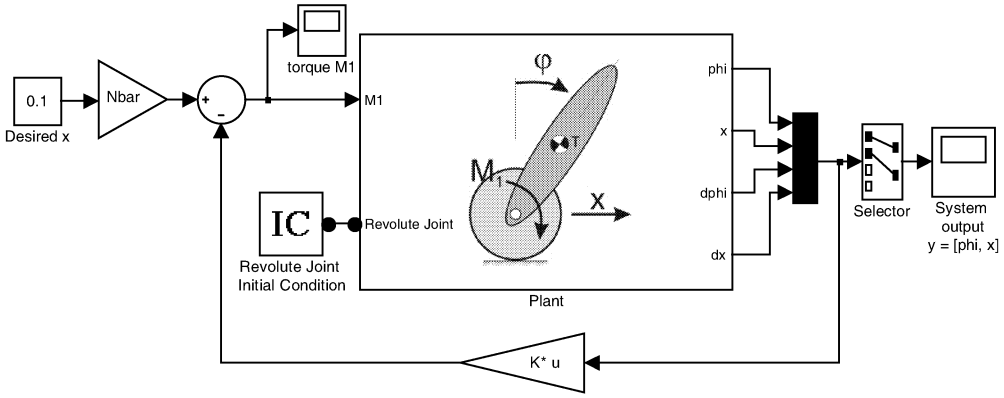

Fig.11: Model of system in Simulink (plant modelled in SimMechanics, testing LQR control using full state vector

An example of system response is shown in Fig. 10. Note that this result includes the influence of sensory processing and dynamics of linear state space observer.

6. Simulation model

After the modelling of plant, sensors and the design of controller, we can improve our model by adding the noises and delays expected in the real system. Noises on accelerometers and delays due to filtering and processing has significant (negative) effect on the quality of the control. Also, the computation time of feedback linearization expressions is crucial.

Next, we should include discrete aspects for some parts of model instead of continuous approach (e.g. the observer and feedback linearization matrixes are to be computed discretely using the microcontroller).

Further, we also add the motor torque limitation.

Finally, we have to treat the datatypes used in Simulink model to follow the requirements and limitation of supposed hardware.

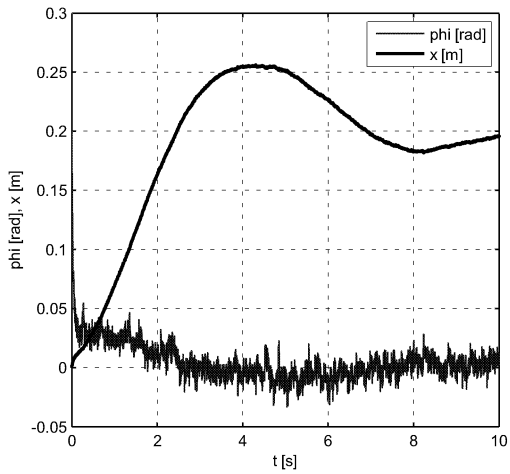

Fig.12: Response of system controlled by feedback linearization (with sensory processing delay 0.8 ms and sensor noise)

The example of system response is shown in Fig. 12.

The natural integration of automatically build mechanical model into the Simulink environment allows a fully parametric design and so the stochastic optimization or evolutionary method application is possible.

7. Testing on physical model

Theoretically achieved results have been verified on the physical model of proposed wheeled balancing device (see Fig. 13).

The control algorithm remains implemented in the Simulink environment. The connection between the computer model and reality is realized by means of special card MF 624 and Real Time Toolbox for Matlab with respective drivers. Sensory data (encoder and dual accelerometer) are read by built-in encoder and analog channels. The DC motor is controlled by PWM voltage converter with power electronics.

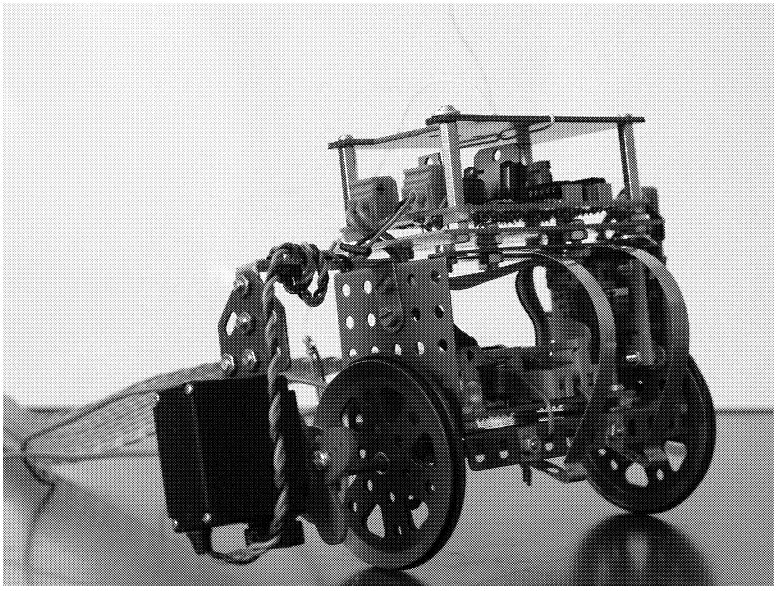

Fig.13: Experimental physical model controlled using Matlab Real Time Toolbox

8. Conclusion

In this paper, we have presented an effective and pragmatic approach to the modelling and control of balanced wheeled inverted-pendulum like robot.

First, there the advantage of automatic dynamic model derivation using Matlab-SimMechanics is shown. Feedback linearization method requires the dynamical model in analytical form and SimMechanics can be employed as the reference and verifies the accuracy of obtained equations. This useful approach can significantly decrease the time required for controller design.

Further, we described the method for computational accelerometer data processing. As a result, we directly obtain the tilt angle. The proposed method is based on the iteration

solution of nonlinear algebraic equation, and with the mentioned adaptation, this method can be successfully used in real-time control of plant (0.8 ms as a delay required for computation).

Next, we designed a complex simulation model in Simulink environment. The plant is modelled using SimMechanics blocks. The sensor noise and delays as well as delays due to feedback linearization computation are added to the model. All parameters of model (plant, control, noises and delays) can be changed from Matlab interface and therefore the arbitrary optimization or automatic testing can be performed.

Two control approaches have been tested: state space control with the K matrix designed by LQR and feedback linearization method. For both of them, the linear (state space) Luenberger observer has been adopted.

Finally, the experiments performed using the physical model have proved the applicability of above described approach. The HIL (hardware in the loop) concept has been applied. The computer control model implemented in Simulink environment has been connected to the real world by means of MF624 data acquisition card with appropriate simulation blocks of Realtime Toolbox for Matlab.

Acknownledgements

Published results were acquired with the support of projects MSM 0021630518 'Simulation modelling of mechatronic systems'.

References

[1] Spong M.W., Hutchinson S., Vidyasagar M.: Robot modeling and control, Wiley, 2006
[2] Grasser F., D'Arrigo A., Colombi S., Rufer A.: JOE: A Mobile, Inverted Pendulum Laboratory of Industrial Electronics, Swiss Federal Institute of Technology Lausanne
[3] Sciavicco L.: Modelling and Control of Robot Manipulators, 2004
[4] Carnegie Mellon University, Control tutorials for Matlab and Simulink, http://www.library.cmu.edu/ctms/ctms/state/state.htm
[5] Wood G.D.: Simulating mechanical systems in Simulink with SimMechanics, The MathWorks Inc., www.mathworks.com, 2002
[6] Tasora A.: An optimized lagrangian-multiplier approach for interactive multibody simulation in kinematic and dynamical digital prototyping, VIII ISCSB, International Symposium on Computer Simulation in Biomechanics, 4–6 July 2001, Politecnico di Milano, Italy, 2001
[7] Caracciolo L., De Luca A., Iannitti S.: Trajectory Tracking Four-Wheel Differentially Driven Mobile Robot, International Conference on Robotics & Automation, Detroit, Michigan, 1999
[8] Spong M.W., Corke P., Lozano R., Nonlinear control of the Reaction Wheel Pendulum, Automatica, pp. 1845–1851, 2001
[9] Reza Olfati-Saber: Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles, PhD thesis, Massachusetts Institute of Technology, 2001
[10] Nguyen H. G., Morrell J., Mullens K., Burmeister A., Miles S., Farrington N., Thomas K., Gagee D.W.: Segway Robotic Mobility Platform, SPIE Proc. 5609: Mobile Robots XVII, Philadelphia, PA, October 27–28, 2004
[11] Davoudi M., Menhaj M.B., Davoudi M.: Motorized Skateboard Stabilization Using Fuzzy Controller International Conference 9th Fuzzy Days in Dortmund, Germany, Sept. 18–20, 2006
[12] Eghtesad M., Necsulescu D.S.: Experimental study of the dynamic based feedback linearization of an autonomous wheeled ground vehicle, Robotics and autonomous systems 47, pp. 47–63, Elsevier, 2004

[13] Ooi R. C.: Balancing a Two-Wheeled Autonomous Robot, The University of Western Australia School of Mechanical Engineering, thesis, 2003
[14] Baloh M., Parent M.: Modeling and Model Verification of an Intelligent Self-Balancing Two-Wheeled Vehicle for an Autonomous Urban Transportation System, The Conference on Computational Intelligence, Robotics, and Autonomous Systems, 2003
[15] Grepl R., Hrabec J.: Virtual prototype of fast differentially driven mobile robot aimed at control algorithm development, Modelling and optimization of physical systems, ISBN 9788360102466, Gliwice, pp. 55–64, 2007
[16] Šolc F., Hrabec J.: Modelling of Fast Differentially Driven Mobile Robot, In New trends in System Simulation, Ostrava: MARQ 2006, pp. 24–29, ISBN: 80-86840-06-9, 2006

Received in editor's office: March 2, 2009
Approved for publishing: March 23, 2009