UNSYS digital

International Journal of Robotics and Mechatronics

Stability Verification of a Control System using the Reachability Analysis

Humaira Salman[†], Farid Gul[§], and Mian Ilyas Ahmad[†]

[†]Research Centre for Modeling and Simulation, National University of Sciences and Technology, Islamabad, Pakistan

[§]School of Electrical Engineering and Computer Science, National Univ. of Sciences and Technology, Islamabad, Pakistan

***Abstract*—**Model checking is a formal method technique used for the verification of safety critical and/or complex software and hardware systems. It provides accurate results by exhaustively exploring the abstract mathematical model of the system. This paper discusses the framework proposed for the verification of an underactuated inverted pendulum based two parallel wheeled vehicle. The physical model of the perceived system is designed for 2 DOF that is the pitch along x-axis and roll along y-axis. The mathematical model of the system is developed using the state-space approach and the system is analyzed using Matlab. Stability is introduced in to the system using a Linear Quadratic Regulator (LQR) with an Observer. This controlled system is analyzed and formally verified using the SpaceEx model checker. The stability verification of this system is performed using the reachability analysis. The model checker results shows the stability verification of the system over the convergence of infinite trajectories for a range of input values provided, showing advantage over simulation based techniques. The results also provide the phase portraits of the output variables in 2-D and 3-D planes.

***Keywords*—**Linear Quadratic Regulator, observer, model checking, reachability, exhaustive search, formal method.

I. Introduction

SAFETY-CRITICAL systems or systems with high complexity need precision and accuracy in their designs. Their precision and accuracy is the foremost concern in the engineering and scientific society. These systems are usually used in fields like medical, transportation and chip design and fabrication etcetera. Design errors in such systems could lead to the implementation of faulty systems and could cause quite a loss including human lives. Therefore, the traditional analysis techniques, which also include simulation based techniques, in such domains could not be relied upon. Formal methods [2-4], however, provide safe, precise and reliable analysis techniques for such systems, overcoming the limitations of the classical analysis and verification techniques such as simulation and testing.

Formal Methods could be better defined as the application of applied mathematics, or in our situation, we could say the application of formal logic to systems for their analysis and design verification. They are fault avoidance and removal techniques used for removing errors and bugs introduced in to the system during the system design. Therefore, they complement techniques, like testing and simulations for fault removal. Formal analysis results are more reliable and accurate than the results obtained from traditional simulation based software, as they suffer from discretization and round off errors. The analysis results are exhaustive compared to simulation and numerical methods based results, as the whole state space of the system could be explored by formal methods, whereas other software could only explore the system performance criteria for a certain points in the state space of the system. Formal methods provide proof of each step like in theorem proves [5-7]. They provide precise and reliable analysis results for complex systems, whose complexity provides technical and operational benefits. They could even be relied upon to give away accurate results for those systems whose stability alters when they are exposed to a different environment compared to the one they were designed in originally. Formal methods provide the evidence for considerations in the design and assessment of safety-critical systems.

Model checking [8, 9] is a formal methods analysis technique. It has been successfully used for the precise analysis of hardware and software systems, for the past few decades. Model checking is a tool that automatically verifies hardware and software-based systems. It has proven to be a successful method to uncover well-hidden bugs in sizeable industrial cases [9]. The applicability of model checking in automated control systems is being however limited. Model checking has not been highly considered for the formal analysis of control systems previously. In the most recent past, some model checking tools have been developed for the analysis and verification of continuous and hybrid control systems [10-13]. Therefore, for this paper, the verification of control systems through model checking is considered as a primary motivation, to emphasize and analyze the ease of using model checking for such systems. Model checking is a computer algebra based system that performs mathematical computations using symbolic algorithms. Also this feature makes it better than the traditional simulation based systems that basically develops a discretized model of the system and simulate the system's output for different input values. Model checking verifies a system by verifying certain properties against it. Generally there are some

Corresponding author: Humaira Salman (e-mail: humaira.gul@gmail.com).
This paper was submitted on February 02, 2014; revised on December 16, 2014; and accepted on December 18, 2014.

basic properties of a system to help analyze and verify it for different performance criteria. Examples are the Safety property, the Reachability property, the Liveness property and the deadlock freeness [8, 9] etcetera. The property of our particular interest in this paper is the reachability property and we will be verifying our system using this property, as. This paper further elaborates the work of [2], that is designing and formally verifying a transportation vehicle's control system, using SpaceEx model checker [11, 12] as shown in the block diagram of **Figure 1**.

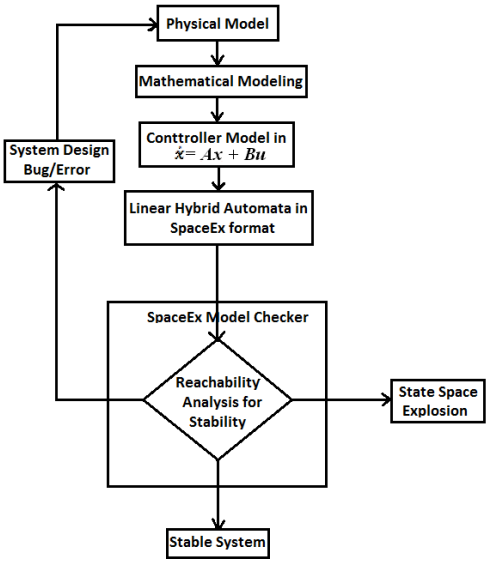


Figure 1 Paper Model

Inverted pendulum based systems has been a major focus of research in the automation and control area for a decade, especially like in the robotic system's mobility designs. The inverted pendulum concept introduced a new concept of balancing and control in Robotic and Mechatronic based mobility systems. The understudy system perceived for the formal analysis is a dynamic, inverted pendulum principle based, coaxially parallel two-wheeled human transportation vehicle. It is an automatic balancing vehicle. Such systems are driven by electric propulsion and a Drive-by-Wire based mechanism controls its dynamics. The system balances on two wheels whether in state of rest or in motion. When at rest, the controlled system tends to remain in a vertically upright position, with the pendulum's angle at 0^0 theoretically. When external force is applied on the pendulum stick in the forward direction, the vehicle starts moving in the forward direction. This happens because the controller commands the actuators to apply torque on the wheels to move the vehicle in forward direction. This balances the pendulum/vehicle from falling and moves the vehicle forward with uniform velocity. It is just like a person holding and balancing a stick on his index finger. Among many examples of such systems on commercial and defence level are the SEGWAY [1], NASA's Robonaut [14] and the V.E.C.N.A's BEAR project [15]. Also on non–commercial level some work has been done on this system, such as the Legways [16] and the Ballbot [17] etc.

A. *Literature Review*

This section provides a brief over view of the literature reviewed for the system design. It was very much helpful and useful in understanding the preliminary design process.

- *SEGWAY HT/PT*

This intellectual idea of a personal transportation vehicle based on the inverted pendulum concept first came from the inventor and entrepreneur, Dean Kamen in Dec 2001. The product was named Segway. It was the first, two coaxially parallel wheeled transportation vehicles that could balance themselves with or without a rider. As it is a commercial transportation vehicle, so the control algorithms are kept as a company trade secret. The vehicle is designed to balance itself both in to an upright vertical position with an angle of 90^o, or it could keep itself in an equilibrium position while in motion with a constant velocity (theoretically). It consists of a standing platform between the two coaxial parallel wheels. The pendulum stick/bar has handles, for the rider to hold that protrudes up from the platform. Along with an assembly of tilt sensors and actuator coupled with the controllers that provide a robust control and regulates balancing to prevent it from falling. This vehicle could move up at a speed of 12.5mph. The vehicle combines expertise in dynamic stabilization using the sensors and actuators to sense the tilt of the pendulum stick of the vehicle and balance it back in to equilibrium position. It is driven by Electric-Propulsion and a DBW (Drive-by-Wire) based system controls its dynamics. This Drive-by-Wire mechanism makes it capable of using actuators and emulators to control the Segway ride, rather than using mechanical parts/components like breaks, pumps and valves etcetera to control the vehicle. This makes the vehicle to provide a better ride with comfort, ease and efficiency. Such an inverted pendulum based moving robot is by default an unstable, multivariable system by nature. It is rather an under-actuated system. From the mechatronics point of view, the system is a complicated design of control and sensor integration [18-20].

Figure 2 Segway HT/PT [1]

Segway (**Figure 2**) has been used in games like Golf and in Polo fests. It is also being used by police force for regular surveillance in shopping plazas, public transportation platforms like airport and railway sub stations. Cameramen use it while recording or filming live matches like Cricket or various other

sport tournaments. Segway has been used in metropolitan areas because it is effective enough for short distances. It is also being used in theme parks like Disney land, San Francisco Bay in California and in Angel Island State Park. And as being told earlier, it uses electric propulsion, therefore offers emission free green transportation.

- *JOE: The Grasser Model*

A research team of the Industrial Electronics Lab at the Swiss Federal Institute of technology, Lausanne, Switzerland, presented the idea and built the prototype of a scaled down model of a moving inverted pendulum on two parallel wheels.

Figure 3 JOE: A mobile inverted pendulum

The model is a digital signal processor controlled vehicle (**Figure 3**) with weight attached at the pendulum stick for simulating a driver. The weighted robotic vehicle has linear controllers decoupled to stabilize the system to equilibrium position. The controllers use information from gyroscopes and encoders attached to the motors, for regulating the stabilization. The research team designed this system by using a fixed weight to simulate the driver, leading to cost reduction and the fatigue removal for using test pilots. This system was made using modern state space theory rather than using the classical control for design, for helping maintain the better control of vehicle speed, angular position and turn-rate of the mobile vehicles. Also this helped in eliminating many variables for the mathematical model design. From the physical model, a mathematical model was derived using state space technique. The main focus of the system control was later on the pitch control and the yaw control. The pitch control is related to the vehicle's vertically upright balance regulation. The yaw control is related to the vehicle's turn rate. The torque applied to the wheel's motor was the only input to this system. As both the pitch control and the yaw control requires this input, so the two control systems were decoupled to overcome this problem. This decoupling made the control of the yaw and pitch control independent of each other for controlling the turn rate and linear speed commands. It also made the troubleshooting of the system much easier.

- *EDGAR*

EDGAR (see **Figure 4**) stands for Electro Drive Gravity Aware Ride [18]. It was an honor's project at the University of Adelaide, Australia, that covered the design of an inverted pendulum based transportation vehicle and it's testing through prototyping. It was a model inspired from SEGWAY itself. The design of EDGAR is based upon those several successes and failures of other attempted designs to replicate SEGWAY. The vehicle designed was robust and easy to use and the strength and proper weight of the vehicle was also insured.

Figure 4 EDGAR [18]

The mathematical model was developed for both with rider transportation vehicle system and a without rider transportation vehicle system. The controller designed for the system considered controlling the pitch, yaw and the roll of the vehicle.

B. *Paper Overview*

Section 2 starts with the design of a dynamic physical model and then on its basis, the design of a mathematical model of a simplified version of the system. This vehicle could move in 2-D which is the Rotation about the pitch axis (lateral axis) and translational motion in the forward/backward direction only to simplify the design and helped us use a fewer variables in the design. This could help reduce the complexity of the model, number of sensors used and cost of the system implementation too. Thus enable us to design the model with 2-DOF.

A controller designed for this system for regulating the stabilization is later presented in Section 3. A Linear Quadratic Regulator with Observer design is discussed in this section for the system. This controller is used to stabilize the system in to equilibrium position.

As the last step of the research, the Formal Verification of the system is performed on this system using Model Checking technique. This formal verification results and details would be shown in Section 4. The system is modeled first according to the model checker requirements for verification, and then the system is checked against the reachability property [13]. Reachability analysis is one of the few features of a model checker which is used to verify a system for certain

performance criteria. In this paper, the reachability property is used for verifying the stability of the system.

C. *Motivation and Contribution*

Although various other software and hardware based systems has been formally verified for performance since the last 2 decades. Yet the formal verification of control systems has not been a very common practice throughout this time. The verification through model checking and control systems has been considered two different domains previously. Model checking as formal verification tool for control systems has not been considered well. But in the very near last few years, some model checking tools have been developed to verify control system as well. Therefore for this research the verification of control systems through model checking was considered as a primary motivation, to emphasize and analyze the ease of using Model Checking for verification of such systems. This research work is the first ever formal verification of such a transportation vehicle using model checking technique.

Inverted pendulum based designs are the basis of modern robotics and stability platforms for different engineered systems. Like in missiles, robots and different transportation vehicles the concept of inverted pendulum is used for their stability over an equilibrium position. Therefore this two wheeled transportation vehicle was chosen to understand the design of such a system and to illustrate a methodology for verifying and insuring the stability of such a dynamic and complex transportation vehicle system by formally analyzing the state space equation based model of the system using the model checker.

II. TRANSPORTATION SYSTEM MODELLING

In this section, system modeling and design of two-wheel transportation vehicle is presented. Such an inverted pendulum based moving robot is by default an unstable system by nature. It is rather an under-actuated system. Modeling and design of a two wheeled transportation vehicle has been a research topic for a while for many researchers either as thesis or dissertation focuses or as a funded research project or even as a journal paper topic [19, 21-24]. The physical and mathematical modeling and the controller design of this system is presented in detail in the section below.

A. *Physical Systems Modeling and Specifications*

Figure 5 and **Figure 6** show the free body diagrams of the chassis and the wheels of the system. **Figure 5** shows the variables of interest of the system and the reaction forces of the wheels on the chassis while **Figure 6** shows the action forces on the wheels by the chassis. The model shows similarities to the model in JOE: A Mobile Inverted Pendulum, and Son of Edgar; State-Space Control of Electro-Drive Gravity-Aware Ride [18, 19]. Except that the Centre of gravity for the model is taken as fixed and is presented above the chassis platform. From **Figure 5** and **Figure 6**, the position and average linear velocity of the vehicle along the x-axis is given by x_W and $\dot{x}_W$. The pendulum stick of the vehicle rotates around the lateral axis, which is the z-axis, by angle θ_P, and angular velocity $\dot{\theta}_P$.

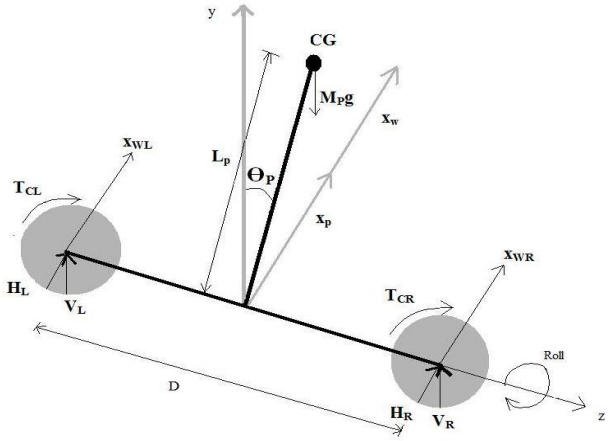


Figure 5 Chassis free body diagram [2]

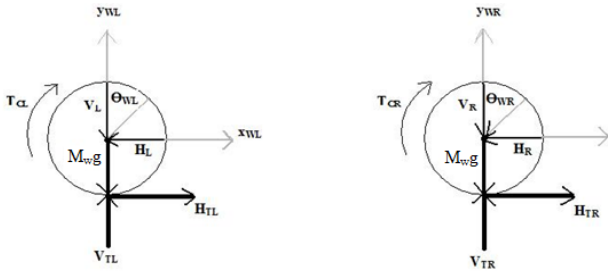


Figure 6 Wheels free body diagram [2]

TABLE. I VARIABLES DESCRIPTION [2]

Variables	Description of the variables
C_T	Translational damping in the system
C_R	Rotational damping in the motors
D	Distance b/w the two wheels centres
d	Duty cycles sent to the left & right motors
g	Acceleration due to gravity (taken as 9.8m/s^2)
H_L, H_R	Horizontal reaction forces of the left and right wheels and the chassis
H_{TL}, H_{TR}	Horizontal tangential force exerted on the left and right wheels by earth/ground
J_P	Moment of inertia of chassis (including the non-rotating parts) along the z-axis
J_W	Moment of Inertia of Chassis (including the rotating parts like shaft etc.) along the z-axis
K_M	Torque constant
L_P	Length of the pendulum from the axle line to the CG
M_P	Chassis mass
M_W	Mass of the rotating parts either of the wheels
R	Terminal resistance of the motors
R	Radius of the wheel
T_{CR}, T_{CL}	Control torques to the left and right motors
V_L,V_R	Vertical reaction forces b/w both the wheels and the chassis
x_p	The pendulums linear displacement
V_{TL}, V_{TR}	Vertical tangential force on left & right wheels by earth/ground

The vehicle is controlled and balanced back to its vertically upright position, or regulated to move with linear forward velocity, by sending duty cycles to the left and right wheel motor's controllers, regulating the amount of sent current and so is the controlled torque produced. ***d*** is the duty cycle sent as an input to the left and right wheels. The controlled torque produced at the left and right wheel motor is given by T_{CR} and T_{CL}. The horizontal and vertical reaction forces are H_L, H_R, V_L and V_R. The centre of gravity, CG, is taken as fixed.

B. *Mathematical Modeling of the System*

The mathematical model design shows the reaction of the system to the given input. State Space approach is used to derive the mathematical model. The parameters and configurations to be taken care of are given below.

- The system basically consists of two parallel coaxial wheels connected to a platform, meant for allowing a human rider to stand on it or for placing load on it in case of an autonomous robotic system.
- The systems straight vertical position may vary by $\pm 5^0$. It would still be considered at 0^0 within this error range.
- The surface is even and the vehicle moves smoothly over it.
- The defects are neglected.
- The vehicle is neither loaded nor ridden.
- Value of input duty cycle, *d*, sent to both the left and right wheel is the same.
- The overcoming obstacles are ignored.
- Effects of aerodynamic resistances are ignored.
- The system has 2-DOF.

Now from the free body diagrams in the above **Figure 5** and **Figure 6**, the balance equations for the system were derived, given as follows.

Balance equation for the left wheel.

$$\Sigma F_X = 0 \Rightarrow \ddot{x}_{WL} M_W = H_{TL} - H_L - C_T \dot{x}_{WL} \quad (1)$$

$$\Sigma F_y = 0 \Rightarrow \ddot{y}_{WL} M_W = V_{TL} - V_L - M_W g \quad (2)$$

$$\Sigma \tau = 0 \Rightarrow \ddot{\theta}_{WL} J_W = T_{CL} - H_{TL} R \quad (3)$$

Balance equation for right wheel.

$$\Sigma F_X = 0 \Rightarrow \ddot{x}_{WR} M_W = H_{TR} - H_R - C_T \dot{x}_{WR} \quad (4)$$

$$\Sigma F_y = 0 \Rightarrow \ddot{y}_{WR} M_W = V_{TR} - V_R - M_W g \quad (5)$$

$$\Sigma \tau = 0 \Rightarrow \ddot{\theta}_{WL} J_W = T_{CR} - H_{TR} R \quad (6)$$

Chassis balance equation:

$$\Sigma F_X = 0 \Rightarrow \ddot{x}_P M_P = H_L + H_R \quad (7)$$

$$\Sigma F_y = 0 \Rightarrow \ddot{y}_P M_P = V_L - M_P g + V_R \quad (8)$$

$$\Sigma \tau = 0 \Rightarrow \ddot{\theta}_P J_P = (H_R + H_L) L_P \mathrm{Cos}\theta_P - (V_R + V_L) L_P \mathrm{Sin}\theta_P - (T_{CL} + T_{CR}) \quad (9)$$

Where $\theta_p \approx 0$, such that $\mathrm{Cos}0 = 1$ & $\mathrm{Sin}\theta_p \approx \theta_p$.

The State space equation derived from the $\dot{x} = Ax + Bu$ form is given below.

$$\begin{bmatrix} \dot{x}_w \\ \ddot{x}_w \\ \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & A_{23} & A_{24} \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ B_{21} \\ 0 \\ B_{41} \end{bmatrix} d \quad (10)$$

And after putting in the values of the system variables, the below mathematical equation was obtained:

$$\begin{bmatrix} \dot{x}_w \\ \ddot{x}_w \\ \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.9422 & -4.28 & 0.03146 \\ 0 & 0 & 0 & 1 \\ 0 & 5.485 & -11.54 & -1.525 \end{bmatrix} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ 0.2402 \\ 0 \\ -11.64 \end{bmatrix} d \quad (11)$$

The terms of the state space equations obtained in equation (10) are given as follows.

$$A_{22} = \{1/[1 + 2[(J_W / R^2) + M_W][1/ M_P + L_P^2/ J_P]\} \times \{-(2K_M^2)/(\mathrm{r}R M_P) - (2K_M L_P^2)/(\mathrm{r}R J_P) - (2K_M^2 L_P)/(\mathrm{r}R J_P) - (2C_T)/M_P - (2C_T L_P^2)/J_P\}$$

$$A_{23} = \{1/ [1 + 2[(J_W/ R^2) + M_W] [1/ M_P + L_P^2/ J_P]\} \times \{-(M_P g L_P)/J_P\}$$

$$A_{24} = \{1/[1 + 2[(J_W / R^2) + M_W][1/ M_P + L_P^2/ J_P]\} \times \{(2 K_M^2)/(\mathrm{r} M_P) + (2 K_M^2 L_P^2)/(\mathrm{r} J_P) - (2 K_M^2 L_P)/(\mathrm{r} J_P) + (2C_T)/(M_P) + (2C_T L_P^2)/ J_P - (2C_T L_P)/J_P\}$$

$$B_{21} = \{1/[1 + 2[(J_W / R^2) + M_W] [1/ M_P + L_P^2/ J_P]\} \times \{(K_M V_S)/(\mathrm{r}M_P) + (K_M V_S L_P^2)/(\mathrm{r}J_P) - (K_M V_S L_P)/(\mathrm{r}J_P)\}$$

$$A_{42} = \{1/ L_P + (2 J_W)/ (R^2 L_P M_P) + (2 M_W)/(L_P M_P)\} \times \{1/[1+2(J_W / R^2 + M_W)(1/ M_P + L_P^2/J_P)]\}\{-(2K_M^2)/(\mathrm{r}R M_P) - (2K_M L_P^2)/(\mathrm{r}R J_P) - (2K_M^2 L_P)/(\mathrm{r}R J_P) - (2 C_T)/ M_P - (2 C_T L_P^2)/ J_P\} + \{(2 K_M^2)/(\mathrm{r} R^2 L_P M_P) + (2C_R)/(L_P M_P R^2) + (2C_T)/(L_P M_P)\}$$

$$A_{43} = \{1/L_P + (2J_W)/ (R^2 L_P M_P) + (2M_W)/ (L_P M_P)\} \times \{1/[1 + 2[(J_W/R^2) + M_W][1/ M_P + L_P^2/ J_P]\} \times \{-(M_P g L_P)/J_P\}$$

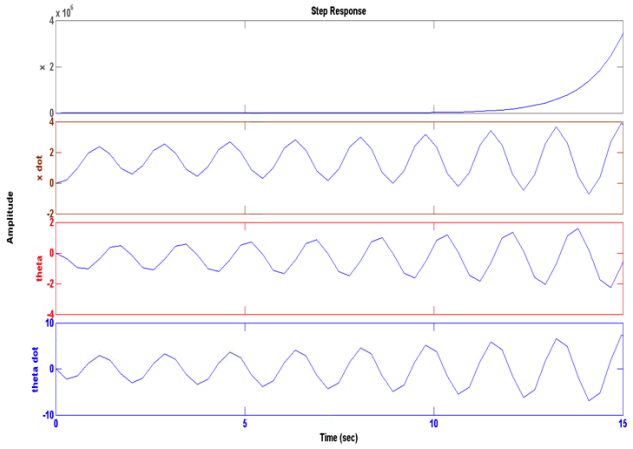
$$A_{44} = \{1/L_P + (2J_W)/ (R^2 L_P M_P) + (2M_W)/(L_P M_P)\} \times \{1/[1 +2[(J_W/ R^2) + M_W][1/ M_P + L_P^2/ J_P]\} \times \{(2K_M^2)/(\mathrm{r}M_P) + (2K_M^2 L_P^2)/(\mathrm{r}J_P) - (2K_M^2 L_P) /(\mathrm{r}J_P) + (2C_R)/(M_P) + (2C_R L_P^2)/ J_P - (2C_R L_P)/J_P\} - \{[2/(L_P M_P \mathrm{R})][(K_M^2/\mathrm{r}) + C_R]\}$$

$$B_{41} = \{1/L_P + (2J_W)/ (R^2 L_P M_P) + (2M_W)/(L_P M_P)\} \times \{1/[1 + 2[(J_W/ R^2) + M_W][1/ M_P + L_P^2/ J_P]\} \times \{(K_M V_S)/(\mathrm{r}M_P) + (K_M V_S L_P^2)/(\mathrm{r}J_P) - (K_M V_S L_P)/(\mathrm{r}J_P)\} - \{(K_M V_S)/(L_P M_P \mathrm{r}R)\}$$

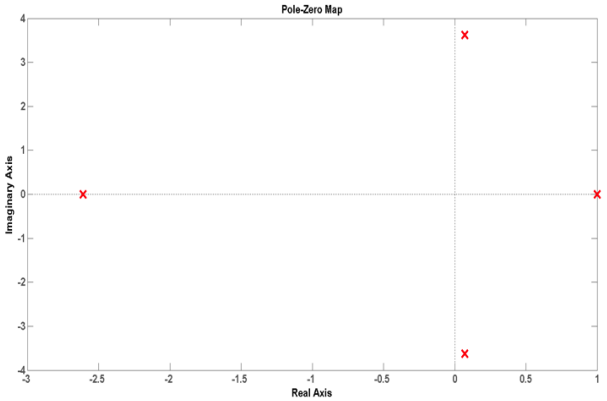
And the output equation for linear velocity and angular position is.

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ \dot{x}_w \\ \theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} d \quad (12)$$

The step response plot and pole-zero maps obtained from the analysis of the system using Matlab, are as shown in **Figure 7**. The poles in **Figure 7 (b)** are on the right side which causes the growing oscillations when the system is exposed to some external disturbances. These results clearly demonstrate the unstable behavior of the system.



(a) Step response of the system model



(b) Pole-zero map of the system model

Figure 7 System analysis results

III. CONTROLLER DESIGN

As the system is unstable as seen from **Figure 7** plots of the previous section, therefore, a controller need to be designed for the system to stabilize the system and balance it back to the equilibrium position. The two main compensator choices for the system are PID and the LQR [25] controller. Both were tried and designed for the system but LQR proved to be a better choice for the system. The PID controller was not chosen for the regulation of our system because our system is a SIMO system, i.e. having a single input and multiple outputs. For the PID control, the state-space equation derived for the system is first to be converted in to the transfer function equation format and for the derivation of two or more than two output equations, we have derive a single output equation at a time while considering and making the rest of the output equations zero. This means we would be compromising on some of the system parameters during the derivation of the output transfer functions. The simulation results might or might not show the proper response plots for the PID controlled system, but the actual system implemented might then suffer instabilities. Therefore, the LQR controller with an observer is a better choice to achieve optimized stability of the system.

A. Linear Quadratic Regulator with Observer Design

The generic aim to design any optimal compensator is to find an admissible time history of control variable $U(t)$, $t \in [t_0, t_f]$ which:

1. Causes the system governed by $\dot{X} = f(t, X, U)$ to follow an admissible trajectory.
2. Optimizes (minimizes/ maximizes) a meaningful performance index J, i.e.
   $J = \varphi(t_f, X_f) + \int L(t, X, U)\, dt$
3. Forces the system to satisfy "proper boundary conditions" [our focus $X(t_0) = X_0$ (given), t_f: fixed]

Our choice of designing an optimal compensator for the system was a Linear Quadratic Regulator (LQR). The objective of the LQR is to derive the state x of a linear (rather linearized) system $\dot{x} = Ax + Bu$ to the origin by minimizing the following quadratic performance index (cost function).

$$J = \int (x^t Q x + u^t R u) dt$$

The LQR method generates a control law, K, to regulate the desired input $u = -Kx$ such that condition 1 is satisfied. If the cost function has different terms, like Q and R here, then the whole idea is to normalize each of the term to contribute fairly in driving the system to the origin while minimizing J.

As our system is an unstable system, and for the transportation model, we need the controlled system to balance back quickly and the damping should die out quickly. So our designed system should be an underdamped system. Similarly the percentage overshoot of 10% and the settling time be somewhat less than 5 seconds.

To start with the design, first the system's controllability and observability is checked. To do so, matlab's commands *rank(co)* and *rank(ob)* is used. Then a full-state feedback was assumed for the LQR design. To compensate the linear displacement (x_w), linear velocity ($\dot{x}_w$), angular position (θ_p) and angular velocity ($\dot{\theta}_p$) of the transportation vehicle, the linear quadratic regulator is used do derive the n×1 control matrix K, and helped minimize the cost function J. For deriving K, the matrix Q, and R were derived first. The weighting factors of the Q matrix and scalar R were chosen by trial and error to reduce the settling time to our desired value. Later a feed forward gain was also introduced and whose value was manually tuned to 3 to reduce the steady-state error. The matrix Q, R and matrix K thus obtained are given as below.

Using $R = 11$,

$Q = C^t \times C$, that is:

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

And

$$K = [3.0151 \;\; 2.7763 \;\; -5.1490 \;\; -1.1511]$$

The LQR design and the matlab simulation for this LQR design is shown in **Figure 8** and **Figure 9**.

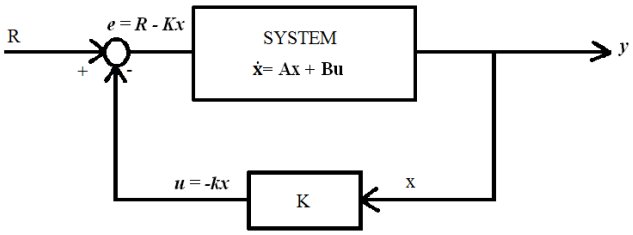


Figure 8 LQR design for the system

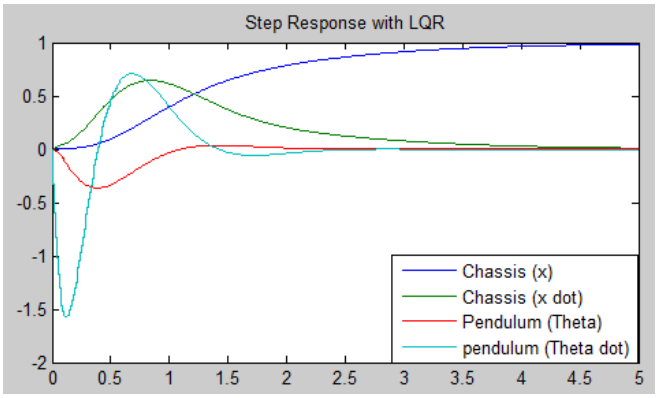


Figure 9 System responses with the LQR

Sensors like gyroscopes and accelerometers are used to measure the output variables of the system, but as these sensors are expensive enough, so to avoid that we use an observer to estimate the states that needs to be measured by sensors. A full-state observer design was then added to estimate the states that are to be measured and was previously only assumed for the control law and LQR design. For the observer design, the poles of the controlled system without the observer were determined. The Observer poles were then placed in position far left of the system poles determined above. Which was intended for the observer based system to work faster than the system without the observer. The controller poles were to be placed away from the imaginary axis because then it would also slow down the system response. The Matlab function *place()* was used for placing the poles. Matrix *L* thus obtained is shown below.

$$L = \begin{bmatrix} 50.4000 & 1.000 & 0 & 0 \\ 0 & 9.3578 & -4.2804 & 0.0315 \\ 0 & 0 & 10.7600 & 1.0000 \\ 0 & 5.4855 & -11.5388 & 4.8754 \end{bmatrix}$$

The eigen values of the system with and without the observer are given as below.

Poles of the regulated	Poles of the regulated
-10.3745	*-10.3745*
-2.5889 + 2.9893i	*-2.5889 + 2.9893i*
-2.5889 - 2.9893i	*-2.5889 - 2.9893i*
-0.9772	*-0.9772*
	-50.4000
	-10.3000
	-10.7600
	-6.4000

The system with observer designed is shown in **Figure 10** and the simulation results shown in **Figure 11**.

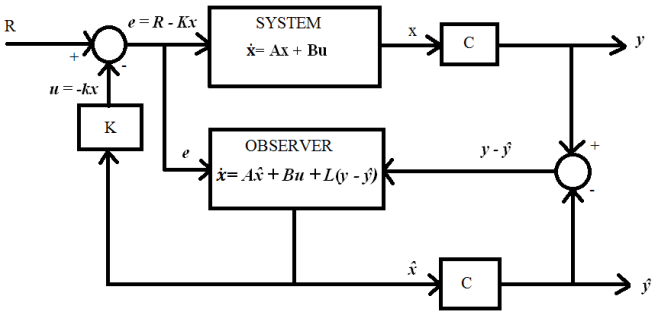


Figure 10 System design of LQR with an observer [2]

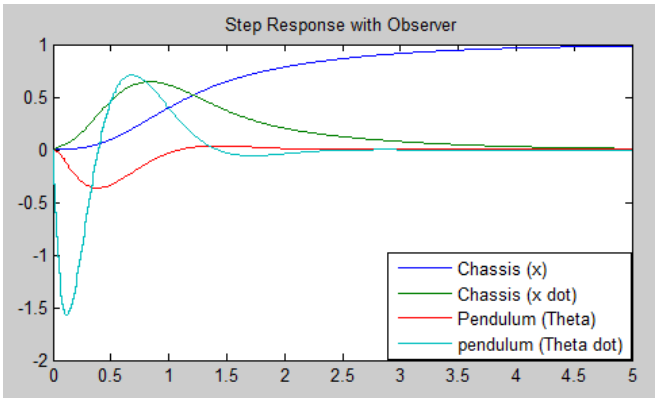


Figure 11 System response with observer/estimator.

The full-state assumption in practical is not a valid idea. So the observer is being designed to estimate the states that were previously assumed and which need to be measured by expensive sensors. The observer estimation is approximately 100% accurate so there is no need for using the sensors for measuring the states. The system with and without an observer showed the same results. Therefore all the system requirements were properly considered in the system design for the two wheeled transportation vehicle. The final state-space matrices obtained for this regulated system are given below.

A =

$$\begin{bmatrix} 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.7241 & -1.6089 & -3.0439 & 0.3079 & 0.7241 & 0.6667 & -1.2366 & -0.2764 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 35.0877 & 37.7938 & -71.4596 & -14.9206 & -35.0877 & -32.3084 & 59.9208 & 13.3960 \\ 0 & 0 & 0 & 0 & -50.4000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10.3000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10.7600 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6.4000 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0.7205 \\ 0 \\ -34.9118 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

These state-space equation matrices were later used for defining the system dynamics for the model checker model.

IV. FORMAL VERIFICATION USING MODEL CHECKING

A. *Model Checking*

Model checking is an automatic verification technique that is applied on systems that could be modeled by automata. In verification through model checking, the first step is to model the system as automata. An automaton is a machine evolving from one state to another under the actions of transitions. Hybrid automata model a system with both discrete and continuous dynamics [9, 26]. Graphically, a hybrid automaton is represented as a labeled directed graph, where nodes represent locations and edges represents discrete transitions. Locations are labeled with differential equations and invariant (stay) conditions and edges are labeled with guards and resets. A symbolic state represents a combination of a location and a variable valuation [9, 11].

After the system is being modeled, model checkers perform the formal analysis on the modeled system by verifying certain property/properties against it. To verify a property for a particular system, the first and foremost point is to identify its application domain for the particular understudy system. To check whether a system is stable, we can verify it by using the reachability analysis over it. If we want that our system should never enter a certain undesired or simply a certain unstable state, we need the safety property to be verified. We want to know whether our stable system could enter some desired/undesired states when faced with some possible environmental/surrounding conditions alteration. For this we use the liveness property. These along the few other properties need to be properly understood before their application to a system's verification.

To describe these properties and to formulate the algorithms, ω-regular and temporal logics [9] has been used widely. Like here, in **Figure 12**, the algorithm and formulae for the room heater example is mentioned and expressed in temporal logic. "Temporal logic is a form of logic specifically tailored for statements and reasoning which involves the notion of order in time [9]".The model checking algorithms verifies whether an automaton satisfies a given temporal formula or not.

B. *Reachability Property*

"The reachability property states that some particular situation can be reached [8, 9]". Like in our case, we have stated the reachability property as stability can be achieved. Or there exists some paths from the initial state of the system, along which stability criteria is satisfied. In our case we verified our perceived system against the reachability property of the system.

The reachability algorithm deals with symbolic states. The algorithm mentioned here performs forward reachability analysis from the initial set and uses a Successor function (Post operator). It traverses a graph while exploring the whole state space and checks whether all the states are reachable or not. The reachability analysis result is the over-approximation of the reachable set.

C. *Hybrid Automata*

For those systems that have both continuous and discrete properties/characteristics in terms of dynamics, events and interactions, they are modeled using hybrid automata [23, 26, 27].

Definition:

The following definition has been adopted from [26].

"An automaton of a hybrid system, called the hybrid automaton H, consists of:

Variables: A finite set $X = \{x_1,\ldots x_n\}$ of real-numbered variables where "n" is called the dimension of H. $\dot{X}$ is written for the set $\{\dot{x}_1 \ldots, \dot{x}_n\}$ where the dotted variables presents the first derivatives during continuous change). And X' is written for the set $\{x_1',\ldots.x_n'\}$ where the primed variables represents values at the conclusion of discrete changes.

Control graph: Also a finite directed multigraph (V, E) where the vertices in V are called control modes and the edges in E are called control switches.

Initial, flow conditions and invariant: The three vertex labeling functions *init*, *inv*, and *flow* which assign three predicates to each control mode $v \in V$. *init(v)* is an initial condition and *inv(v)* is an invariant condition. And each *inv(v)* and *init(v)*, being a predicate, has free variables from *X*. *flow(v)* is a flow condition and it is a predicate whose free variables are from $X \cup \dot{X}$.

Jump conditions: *jump* is an edge labeling function that assigns a predicate to each control switch $e \in E$. *jump(e)* is a jump condition and it is a predicate having free variables from $X \cup X'$.

Events: Σ, a finite set of events, and *E*, an edge labeling function *event:* $E \rightarrow \Sigma$ which assigns an event to each control switch" Henzinger [26].

Figure 12 shows an automaton of a room heater as an example of a hybrid automaton. *Switch_On* and *Switch_Off* are the labels of the two transitions or jumps from the two locations *On* and *Off*. $\dot{T}$ = $rate_{up}$ and $\dot{t}$= $rate_{down}$ are the dynamics or rates or flow of the two locations. $T \geq temp_{on}$ and $T \leq temp_{off}$ are the invariants of these locations. As the guard of the transition *Switch_On* and the invariant of location *Off* overlaps at $T = temp_{on}$, and also $\dot{T} > 0$ in location *Off*. So when the temperature of the heater *T* raises to or above $temp_{on}$ the jump from location *Off* to location *On* occurs. The automaton location remains *On* till the temperature *T* drops below or becomes equal to the threshold temperature $temp_{off}$ of the location. Again if this happens and the guard of the transition Switch_Off and the invariant of the location *On* overlaps, the jump from state *On* to state *Off* occurs.

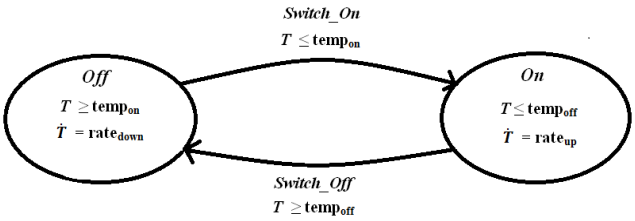


Figure 12 Hybrid automaton of a room heater [2]

For our case, the flow of the two wheeled transportation system designed, is in the form $\dot{x}$=A*x*+B*u*. Where *x* and $u \in R^n$ and A is an n×n matrix.

D. *Formal Analysis using Reachability Property*

Control systems are normally presented by differential equations and they contain infinite number of states. Therefore an approach like Reachability could be used for such continuous systems to verify their stability and safety properties. Control systems are hybrid systems [26]. As mentioned in the previous section, hybrid systems have both discrete and continuous components [13]. There are different model checkers that are used for the automatic verification of hybrid systems. Two among them as an example are mentioned here, i.e. HyTech [15] and PHAVer [28]. Both of these symbolic model checkers verifies linear hybrid automata; a class of hybrid automata that is defined by linear predicates and piecewise constant bounds on the derivatives.

- *SpaceEx Model Checker*

The model checker we have chosen to perform the model checking was SpaceEx [11, 12]. It is a tool for verification of hybrid systems using two different scenarios for verification. It uses both the PHAVer algorithms and the LGG (Le Guernic Gerard) algorithm to verify hybrid systems according to the two scenario's conditions and verification requirements. The PHAVer algorithm applies only to the linear hybrid automaton based systems. The LGG support functions applies to those systems with piecewise affine dynamics having non-deterministic inputs [11].

SpaceEx uses a bit different technique in modeling the system in to a symbolic state automaton and analyzing it. It has three main components to model and analyze the system with: The Model Editor, the Analysis core and the web interface [11]. As mentioned earlier the LGG support function accepts the dynamics of the system of the form given as:

$\dot{x}$=Ax + Bu

As the .sx format in SpaceEx does not accept vector/matric notations. Therefore the above equation could be described in the .sx format as given below.

$\dot{x}_1 == a_{11}*x_1 + ... + a_{1n}*x_n + b_{11}*u_1$&

$\dot{x}_2 == a_{21}*x_1 + ... + a_{2n}*x_n + b_{21}*u_1$&

...

…

$\dot{x}_n == a_{n1}*x_1 + ... + a_{nn}*x_n + b_{n1}*u_1$

Or by an equivalent expression, the equation could also be expressed as:

$x_1 := a_{11}*x_1 + ... + a_{1n}*x_n + b_{11}*u_1$&

$x_2 := a_{21}*x_1 + ... + a_{2n}*x_n + b_{21}*u_1$&

...

…

$x_n := a_{n1}*x_1 + ... + a_{nn}*x_n + b_{n1}*u_1$

Un-explicitly assigned variables are supposed to remain constant in the transition. Guards and invariants are presented in the form of convex linear constraints on the variables, whereas conjunctions are denoted by an ampersand (&). Such as:

$a*x+b*y == 1$ & $c \leq z \leq d$

Universal constraint and un-satisfiable constraints are denoted by true and false [11].

- *Modeling in SpaceEx*

Using the model editor for SpaceEx model checker, the hybrid automaton for the transportation system was created. First the three base components of the automaton were created. Among the three base components, the first one was to model the component representing the system's abstract model with the dynamics of the regulated system, showing the behavior of the system. The dynamics of the system were modeled using the final state-space equations of the system obtained from the system regulated by the LQR with an observer in the previous section. These system equations of the controlled system were modeled in to the SpaceEx automaton format. Among the eight system variables from x_1 to x_8, four are the system variables,

while the remaining four are the controller variables. The linear position is represented by x_1, linear velocity by x_2, angular position by x_3 and angular velocity by x_4. The matrices A and B converted into the system flow equation i.e. the .sx format is as mentioned below.

$$\dot{x}_1 == (0.000000000000)*x_1 + (1.000000000000)*x_2 + (0.000000000000)*x_3 + (0.000000000000)*x_4 + (0.000000000000)*x_5 + (0.000000000000)*x_6 + (0.000000000000)*x_7 + (0.000000000000)*x_8 + (0.000000000000)*u_1 \&$$

$$\dot{x}_2 == (-0.724100000000)*x_1 + (-1.608900000000)*x_2 + (-3.043900000000)*x_3 + (0.307900000000)*x_4 + (0.724100000000)*x_5 + (0.666700000000)*x_6 + (-1.23660000000)*x_7 + (-0.276400000000)*x_8 + (0.720500000000)*u_1 \&$$

$$\dot{x}_3 == (0.000000000000)*x_1 + (0.000000000000)*x_2 + (0.000000000000)*x_3 + (1.000000000000)*x_4 + (0.000000000000)*x_5 + (0.000000000000)*x_6 + (0.000000000000)*x_7 + (0.000000000000)*x_8 + (0.000000000000)*u_1 \&$$

$$\dot{x}_4 == (35.087700000000)*x_1 + (37.793800000000)*x_2 + (-71.459600000000)*x_3 + (-14.920600000000)*x_4 + (-35.087700000000)*x_5 + (-32.308400000000)*x_6 + (59.920800000000)*x_7 + (13.396000000000)*x_8 + (-34.911800000000)*u_1 \&$$

$$\dot{x}_5 == (0.000000000000)*x_1 + (0.000000000000)*x_2 + (0.000000000000)*x_3 + (0.000000000000)*x_4 + (-50.400000000000)*x_5 + (0.0000000000000)*x_6 + (0.0000000000000)*x_7 + (0.000000000000)*x_8 + (0.000000000000)*u_1 \&$$

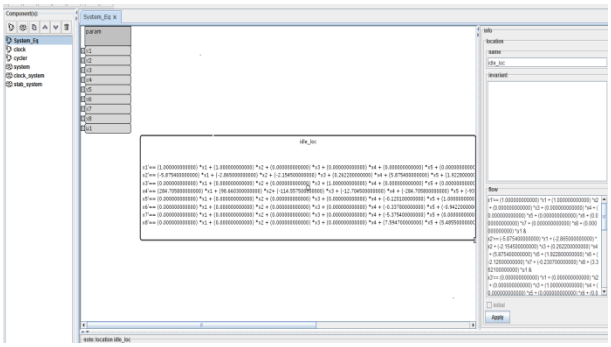
$$\dot{x}_6 == (0.000000000000)*x_1 + (0.000000000000)*x_2 + (0.000000000000)*x_3 + (0.000000000000)*x_4 + (0.0000000000000)*x_5 + (-10.300000000000)*x_6 + (0.0000000000000)*x_7 + (0.0000000000000)*x_8 + (0.000000000000)*u_1 \&$$

$$\dot{x}_7 == (0.000000000000)*x_1 + (0.000000000000)*x_2 + (0.000000000000)*x_3 + (0.000000000000)*x_4 + (0.0000000000000)*x_5 + (0.000000000000)*x_6 + (-10.760000000000)*x_7 + (0.000000000000)*x_8 + (0.000000000000)*u_1 \&$$

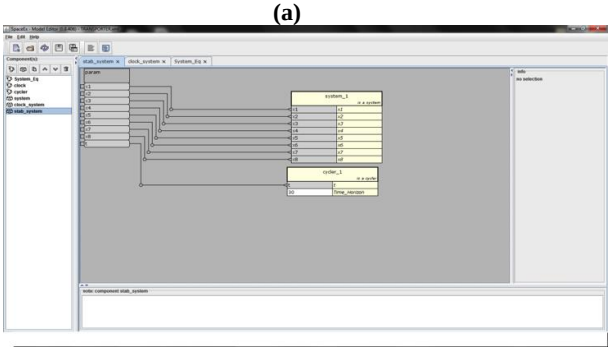
$$\dot{x}_8 == (0.000000000000)*x_1 + (0.000000000000)*x_2 + (0.000000000000)*x_3 + (0.000000000000)*x_4 + (0.000000000000)*x_5 + (0.000000000000)*x_6 + (0.000000000000)*x_7 + (-6.400000000000)*x_8 + (0.000000000000)*u_1$$

After that the other two base components are created to show the time lapse and the location exploration cycles till the reachability time horizon. The three base components were then named System_Eq, Clock_system and Cycler. Then these three components are bound together to model the network components of the automaton. Once the hybrid automaton is created, the .xml file for the system is generated. The .cfg file was created for the specification of the initial conditions of the variables, the output variables, reachability control commands, analysis results scenario and time step values. The XML code for the system modeled is given in Appendix A. This model along with its conditions file is given to the analysis core using the web interface provided by SpaceEx. The system is analyzed using the reachability analysis and the output results are verified for the stability of the system, which are presented in section 4. The reachability of the system in our case is to check whether the stable state is reachable or not. The results obtained for the system are very good and it verifies the correctness of the system designed.

Figure 13 and **Figure 14** shows the model editor snapshots of the hybrid system modeling in the SpaceEx editor and the SpaceEx web interface.



(a)



(b)

Figure 13 System automaton modeling in SpaceEx model editor

Figure 14 SpaceEx web interface

V. ANALYSIS RESULTS AND DISCUSSIONS

A. *Reachability Analysis Results*

In SpaceEx, the results were obtained in text format and also in both 2-D and 3-D graphical formats as well. The results obtained were obtained for the four output variables of the system as required. The scenario selection along with various other commands was provided in the analysis core. The initial condition chosen for the output variables that is the linear position (x_1), linear velocity (x_2), angular position (x_3) and the angular velocity (x_4) is 0.1. **Figure 15-21** shows these results more clearly and shows the flowpipe approximation of these results in SpaceEx.

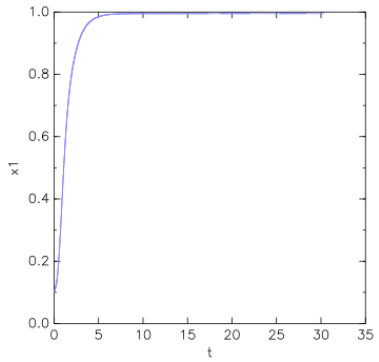


Figure 15 Reachability analysis result of (t, x_1) with initial value of x_1=0.1

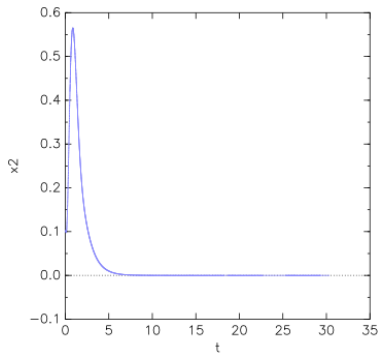


Figure 16 Reachability analysis result of (t, x_2) with initial value of x_2=0.1

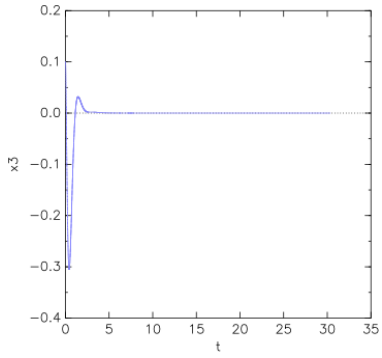


Figure 17 Reachability analysis result of (t, x_3) with initial value of x_3=0.1

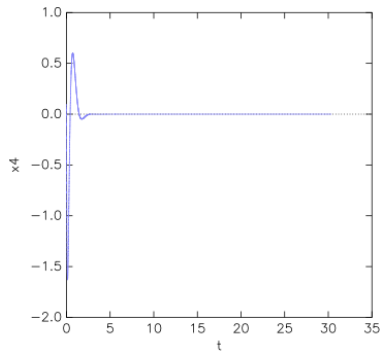


Figure 18 Reachability analysis of (t, x_4) with initial value of x_4=0.1

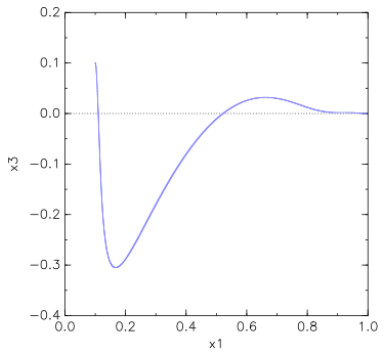


Figure 19 Reachability Analysis phase portrait of (x_1, x_3, t) with an initial value of x_1 and x_3 equal to 0.1

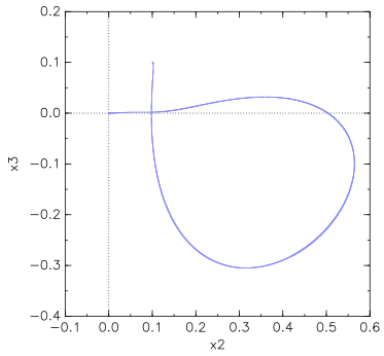


Figure 20 Reachability Analysis phase portrait of (x_2, x_3, t) with an initial value of x_2 and x_3 equal to 0.1

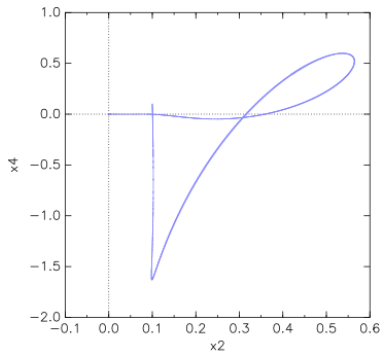
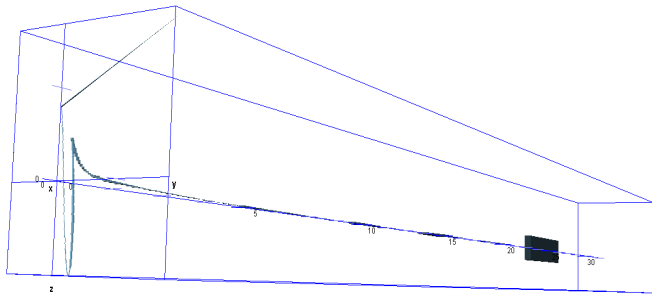
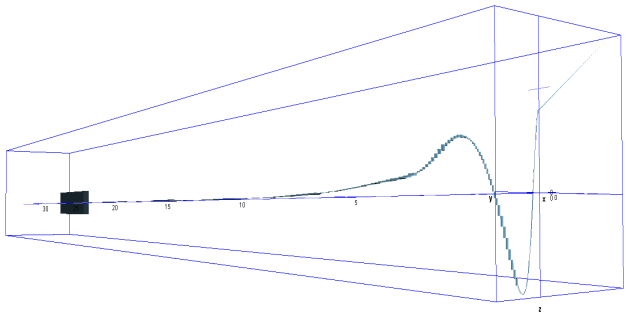


Figure 21 Reachability Analysis Phase portrait of (x_2, x_4, t) with an initial value of x_2 and x_4 equal to 0.1

As it could be seen that the trajectories are evolving from the initial condition and converges to 0. The convergence of these trajectories verifies stability of the designed system. The results of **Figure 19-21** are of particular interest here. It shows the stability behavior of the system variables by showing the phase portrait of the combine trajectories between x_2-x_3 and x_2-x_4, on which they are stabilizing under the given initial conditions. **Figure 22** shows the 3-D graphical presentation of the response for x_1 and x_3 only, in the graphical user interface of SpaceEx model checker. The 3-D results shown for x_1 and x_3 are just to show the capabilities and different interfaces of SpaceEx model checker. The results for the rest of the variables show similar response by converging to 0, therefore they are not shown any further. The variable x_1 is mapped along the x-axis and x_3 is mapped along the y-axis.



(b) 3-D graphical presentation of the verification results for (x_1, x_3, t) with an initial value of x_1 and x_3 equal to 0.1



(b) 3-D graphical presentation of the verification results for (x_1, x_3, t) with an initial value of x_1 and x_3 equal to 0.1

Figure 22 3D graphical presentation of the verification results for x_1 and x_3. (a) and (b) are meant to show the 3-D results from 2 different angles only. The results although are same.

B. *Reachability Analysis for a Range of Initial Values*

In SpaceEx we can also verify and show the stability response for the variables having a range of initial conditions. It could be very helpful to check whether there is any evolving trajectory of the variable for a certain initial value, for which the system shows an unstable response. The verification of a system's variable for such an interval is a time consuming process in simulation based software. It might need infinite (theoretically) simulations to calculate the response for such continuous intervals. One of the benefits of model checking is that it could calculate the evolving trajectories for systems having such intervals in a much lesser time comparatively with high precision. **Figure 23-29** shows the reachability analysis results for the system variables with initial condition of 0.1 except for x_3 which ranges from 0.09 to 0.79, given as $0.09 \leq x_3 \leq 0.79$.

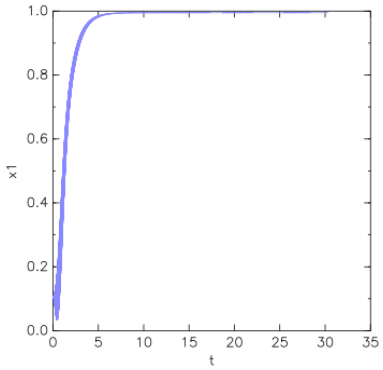


Figure 23 Reachability analysis of infinite (theoretically) trajectories of (t, x_1) for a range of initial values of x_1 i.e. $0.09 \leq x_1 \leq 0.79$

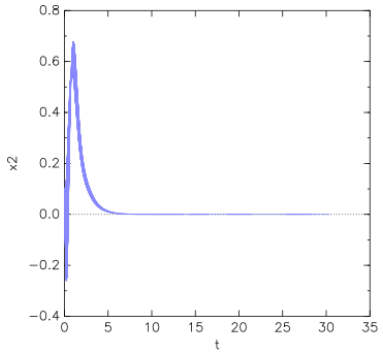


Figure 24 Reachability analysis of infinite (theoretically) trajectories of (t, x_2) for a range of initial values of x_2 i.e. $0.09 \leq x_2 \leq 0.79$

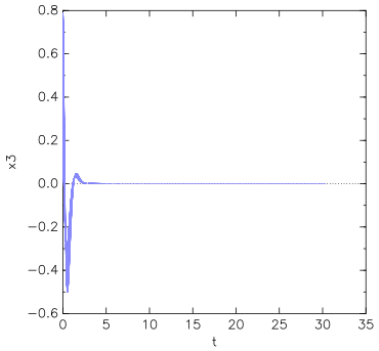


Figure 25 Reachability analysis of infinite (theoretically) trajectories of (t, x_3) for a range of initial values of x_3 i.e. $0.09 \leq x_3 \leq 0.79$

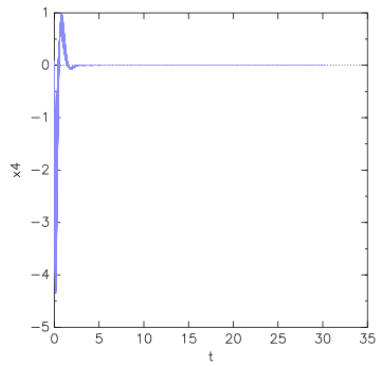


Figure 26 Reachability analysis of infinite (theoretically) trajectories of (t, x_4) for a range of initial values of x_4 i.e. $0.09 \leq x_4 \leq 0.79$

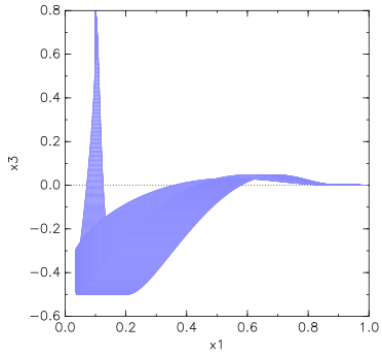


Figure 27 Reachability analysis phase portrait of infinite (theoretically) trajectories of (x_1, x_3, t) for initial value for $x_1=0.1$ and $0.09 \leq x_3 \leq 0.79$

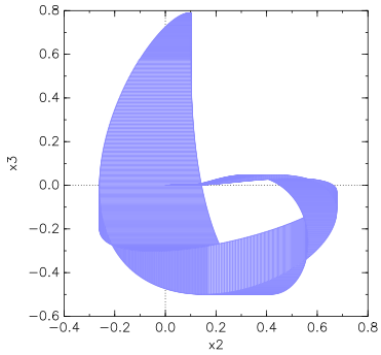


Figure 28 Reachability analysis phase portrait of infinite (theoretically) trajectories of (x_2, x_3, t) for initial value for $x_2=0.1$ and $0.09 \leq x_3 \leq 0.79$

Figure 27-29 results could also be shown in 3-D for further insight. **Figure 30** shows the picture of the 3-D results obtained for x_1 and x_3 only. The initial condition of the variables x_1 is 0.1 and x_3 is lying in the range $0.09 \leq x_3 \leq 0.79$. From **Figure 27-30**, it could be seen that there are no evolving unstable trajectories for the given range of initial values. Hence our system's stability and correctness was verified by formal analysis using model checking techniques. The whole analysis results also shows how the model checking approach based on optimization along with the receding horizon control and the computational resources makes it able to automatically verify the system under consideration with precision, ease and simplicity.

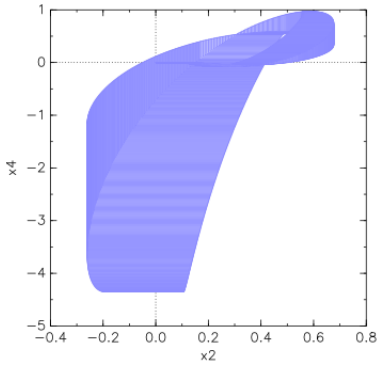
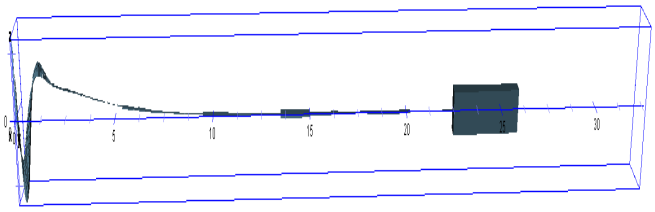
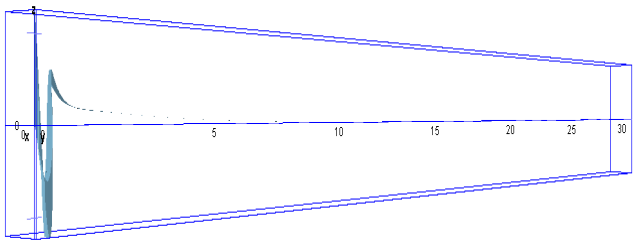


Figure 29 Reachability analysis phase portrait of infinite (theoretically) trajectories of (x_2, x_4, t) for initial value for x_2 and $x_4=0.1$



(a)



(b)

Figure 30 3-D formal analysis result of of infinite (theoretically) trajectories of (x_1, x_3, t) for initial value for $x_1=0.1$ and $0.09 \leq x_3 \leq 0.79$. (a) and (b) are meant to show the 3-D results from 2 different angles only. The results although are same.

VI. CONCLUSIONS AND FUTURE RECOMMENDATIONS

A. *Conclusions*

A dynamic physical model of a co-axially parallel two-wheeled transportation vehicle was created and then from this model, the mathematical model for the system was derived using the state-space approach. After the system analysis results, that showed an unstable system, the system was regulated by designing a linear quadratic regulator for it. An observer is later added to the system and the simulation results for the regulated system, both with and without an observer, are compared, which showed complete similarity. Which means the parameters and specifications for the designed system are properly met.

Later for the formal verification through model checking, the system was modeled in to a SpaceEx hybrid automaton format, with the system dynamics modeled as the flow of the system. Reachability property was verified for the system, which verifies the stability of the system. At the end the reachability analysis results were presented for both the desired output variables. The specific results of interest are those with infinite trajectories (theoretically) for a range of initial values of x_3 and its corresponding response results of the rest of the output variables for that range. The results showed the exhaustive exploration of the whole state-space of the system, starting from that initial range of values. There were no possible unstable trajectories shown for the complete range of values, which shows the proper regulation of the system by the controller.

This paper shows the ease and efficiency of using model checking technique for the formal verification of control systems. Also the exhaustive exploration for the system was completed in a very less time, which is also a huge advantage over simulation based techniques that takes days to analyze a system.

B. *Future Recommendations*

There had been some certain safety issues regarding such transportation vehicles like SEGWAY. These issues were caused due to balancing the vehicle after a certain inclination angle. Some major accidents had been reported since the last few years causing severe injuries to neck, back, facial and traumatic brain injuries as well as fractured arms and legs. This problem has also raised some traffic issues, due to which new traffic laws regarding such vehicles have been implied in different states of USA and in some European countries.

To resolve these issues, Controller Synthesis [10] could be applied after the system verification. The safety analysis [9] could later give the information regarding the bad states of the system. Bad states are those states that lie in the region where our system could go above that certain inclination angle, which can make our proposed system unstable. The controller synthesis could enable us to avoid entering these bad states, thus maintaining system stability. This approach will enable us to stabilize our vehicle even above that certain inclination angle.

At the end, further suggestions could be to design the system for three dimensions and the reachability of the system could again be checked for more than 2 variables, as the addition of the third dimension would add more variables to the system. Also the external disturbances and frictional resistances could be added that were neglected for this system's design.

REFERENCES

[1] G. R. Faulhaber. *SEGWAY INC.* VIEW ITEM

[2] H. Gul, J. Ahmad, F. Gul, and M. Ilyas, "Modeling and formal verification of inverted pendulum based two-wheeled transportation vehicle," in *SICE Annual Conference (SICE), 2012 Proceedings of*, 2012, pp. 113-118.

[3] F. Borrelli, "Constrained optimal control of linear and hybrid systems," 2003.

[4] T. Wongpiromsarn, "Formal methods for design and verification of embedded control systems: Application to an autonomous vehicle," California Institute of Technology, 2010.

[5] B. Mathieu, P. Melchior, A. Oustaloup, and C. Ceyral, "Fractional differentiation for edge detection," *Signal Processing*, vol. 83, pp. 2421-2432, 2003. CrossRef

[6] X. Moreau, C. Ramus-Serment, and A. Oustaloup, "Fractional differentiation in passive vibration control," *Nonlinear Dynamics*, vol. 29, pp. 343-362, 2002. CrossRef

[7] B. Vinagre, I. Petráš, I. Podlubny, and Y. Chen, "Using fractional order adjustment rules and fractional order reference models in model-reference adaptive control," *Nonlinear Dynamics*, vol. 29, pp. 269-279, 2002. CrossRef

[8] C. Baier and J. Katoen, "Principles of Model Checking (Representation and Mind Series). 2008," ed: The MIT Press.

[9] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, *et al.*, *Systems and software verification: model-checking techniques and tools*: Springer Publishing Company, Incorporated, 2010.

[10] E. Asarin, T. Dang, and O. Maler, "The d/dt tool for verification of hybrid systems," in *Computer Aided Verification*, 2002, pp. 365-370.

[11] G. Frehse, "An introduction to spaceex v0. 8," ed: December, 2010.

[12] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, *et al.*, "SpaceEx: Scalable verification of hybrid systems," in *Computer Aided Verification*, 2011, pp. 379-395.

[13] A. J. Van Der Schaft, J. M. Schumacher, A. J. van der Schaft, and A. J. van der Schaft, *An introduction to hybrid dynamical systems* vol. 251: Springer London, 2000. CrossRef

[14] (2010). *SEGWAY ROBOTICS*. VIEW ITEM

[15] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," in *Computer aided verification*, 1997, pp. 460-463.

[16] Steve'sLegway. VIEW ITEM

[17] *BALLBOT Project*. VIEW ITEM

[18] N. Baker, C. Brown, D. Dowling, J. Modra, and D. Tootell, "SON of EDGAR: State-space cONtrol of Electro-Drive Gravity-Aware Ride," Honours thesis, The University of Adelaide, 2006.

[19] F. Grasser, A. D'Arrigo, S. Colombi, and A. C. Rufer, "JOE: a mobile, inverted pendulum," *Industrial Electronics, IEEE Transactions on*, vol. 49, pp. 107-114, 2002. CrossRef

[20] A.-Š. Vitko, Michal - Jurišica, Ladislav - Hubinský, Peter, "Data fusion and context awareness in autonomous robotics," *International Journal of Mechanics and Control*, vol. 5, pp. 29-39, 2004.

[21] C.-H. Chiu, W.-R. Tsai, M.-H. Chou, and Y.-F. Peng, "Two-wheeled robot control based on self-tuning output recurrent CMAC," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009*, 2009.

[22] R. Grepl, "Balancing Wheeled Robot: Effective Modelling, Sensory Processing and Simplified Control," *Engineering Mechanics*, vol. 16, pp. 141-154, 2009.

[23] S. Nawawi, M. Ahmad, and J. Osman, "Real-time control of a two-wheeled inverted pendulum mobile robot," *World Academy of Science, Engineering and Technology*, vol. 39, pp. 214-220, 2008.

[24] A. Gmiterko, Vackova, x, and M., "Dynamic model of vehicle with two coaxial parallel wheels," in *Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium on*, 2011, pp. 303-305.

[25] K. M. Passino and N. Quijano, "Linear quadratic regulator and observer design for a flexible joint," *Department of Electrical Engineering, The Ohio State University*, 2002.

[26] T. A. Henzinger, *The theory of hybrid automata*: Springer, 2000.

[27] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, *et al.*, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, pp. 3-34, 1995. CrossRef

[28] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*, ed: Springer, 2005, pp. 258-273.

VII. APPENDIX A: XML CODE

```
<?xml version="1.0" encoding="iso-8859-1"?>
<sspaceex xmlns="http://www-verimag.imag.fr/xml-namespaces/sspaceex" version="0.2" math="SpaceEx">
<component id="System_Eq">
<param name="x1" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x2" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x3" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x4" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x5" type="real" local="false" d1="1" d2="1" dynamics="any"/>
<param name="x6" type="real" local="false" d1="1" d2="1" dynamics="any"/>
<param name="x7" type="real" local="false" d1="1" d2="1" dynamics="any"/>
<param name="x8" type="real" local="false" d1="1" d2="1" dynamics="any"/>
<param name="u1" type="real" local="false" d1="1" d2="1" dynamics="const" />
<location id="1" name="idle_loc" x="565.0" y="263.0" width="138.0" height="166.0">
<flow>x1'== (0.000000000000) *x1 + (1.000000000000) *x2 + (0.000000000000) *x3 + (0.000000000000) *x4 + (0.000000000000) *x5 + (0.000000000000) *x6 + (0.000000000000) *x7 + (0.000000000000) *x8 + (0.000000000000) *u1 &amp;;

x2'== (-0.724100000000) *x1 + (-1.608900000000) *x2 + (-3.043900000000) *x3 + (0.307900000000) *x4 + (0.724100000000) *x5 + (0.666700000000) *x6 + (-1.23660000000) *x7 + (-0.276400000000) *x8 + (0.720500000000) *u1 &amp;;
x3'== (0.000000000000) *x1 + (0.000000000000) *x2 + (0.000000000000) *x3 + (1.000000000000) *x4 + (0.000000000000) *x5 + (0.000000000000) *x6 + (0.000000000000) *x7 + (0.000000000000) *x8 + (0.000000000000) *u1 &amp;;

x4'== (35.087700000000) *x1 + (37.793800000000) *x2+ (-71.459600000000) *x3 + (-14.920600000000) *x4 + (-35.087700000000) *x5 + (-32.308400000000) *x6 + (59.920800000000) *x7 + (13.396000000000) *x8 + (-34.911800000000) *u1 &amp;;

x5'== (0.000000000000) *x1 + (0.000000000000) *x2 + (0.000000000000) *x3 + (0.000000000000) *x4 + (-50.400000000000) *x5 + (0.0000000000000) *x6 + (0.0000000000000) *x7 + (0.000000000000) *x8 + (0.000000000000) *u1 &amp;;

x6'== (0.000000000000) *x1 + (0.000000000000) *x2 + (0.000000000000) *x3 + (0.000000000000) *x4 + (0.0000000000000) *x5 + (-10.300000000000) *x6 + (0.0000000000000) *x7 + (0.0000000000000) *x8 + (0.000000000000) *u1 &amp;;

x7'== (0.000000000000) *x1 + (0.000000000000) *x2 + (0.000000000000) *x3 + (0.000000000000) *x4 + (0.0000000000000) *x5 + (0.000000000000) *x6 + (-10.760000000000) *x7 + (0.000000000000) *x8 + (0.000000000000) *u1 &amp;;

x8'== (0.000000000000) *x1 + (0.000000000000) *x2 + (0.000000000000) *x3 + (0.000000000000) *x4 + (0.000000000000) *x5 + (0.000000000000) *x6 + (0.000000000000) *x7 + (-6.400000000000) *x8 + (0.000000000000) *u1</flow>
</location>
</component>
<component id="clock">
<param name="t" type="real" local="false" d1="1" d2="1" dynamics="any" />
<location id="1" name="clock_ticks" x="212.0" y="109.0">
<flow>t'==1</flow>
</location>
</component>
<component id="cycler">
<param name="t" type="real" local="false" d1="1" d2="1" dynamics="any" />
<param name="Time_Horizon" type="real" local="false" d1="1" d2="1" dynamics="const" controlled="false" />
<param name="jump" type="label" local="true" />
<location id="1" name="idle_loc" x="263.0" y="232.0" width="174.0" height="150.0">
<invariant>t&lt;=Time_Horizon</invariant>
<flow>t'==1</flow>
</location>
<transition source="1" target="1">
<label>jump</label>
<guard>t&gt;=Time_Horizon</guard>
<assignment>t:=0</assignment>
<labelposition x="-44.0" y="-84.0" width="110.0" height="92.0" />
</transition>
</component>
<component id="system">
<param name="x1" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x2" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x3" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x4" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x5" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x6" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x7" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x8" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<bind component="System_Eq" as="System_Eq_1" x="481.0" y="42.0">
<map key="x1">x1</map>
<map key="x2">x2</map>
<map key="x3">x3</map>
<map key="x4">x4</map>
<map key="x5">x5</map>
<map key="x6">x6</map>
<map key="x7">x7</map>
<map key="x8">x8</map>
<map key="u1">1</map>
</bind>
</component>
<component id="clock_system">
<param name="x1" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
<param name="x2" type="real" local="false" d1="1" d2="1" dynamics="any" controlled="true" />
```

```
<param name="x3" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x4" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x5" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x6" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x7" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x8" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="t" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
    <bind component="system" as="system_1" x="562.0" y="109.0">
      <map key="x1">x1</map>
      <map key="x2">x2</map>
      <map key="x3">x3</map>
      <map key="x4">x4</map>
      <map key="x5">x5</map>
      <map key="x6">x6</map>
      <map key="x7">x7</map>
      <map key="x8">x8</map>
    </bind>
    <bind component="clock" as="clock_1" x="534.0" y="347.0">
      <map key="t">t</map>
    </bind>
  </component>
  <component id="stab_system">
<param name="x1" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x2" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x3" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x4" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x5" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x6" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x7" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="x8" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
<param name="t" type="real" local="false" d1="1" d2="1" dynamics="any"
controlled="true" />
      <bind component="system" as="system_1" x="558.0" y="64.0">
      <map key="x1">x1</map>
      <map key="x2">x2</map>
      <map key="x3">x3</map>
      <map key="x4">x4</map>
      <map key="x5">x5</map>
      <map key="x6">x6</map>
      <map key="x7">x7</map>
      <map key="x8">x8</map>
    </bind>
    <bind component="cycler" as="cycler_1" x="547.0" y="276.0">
      <map key="t">t</map>
      <map key="Time_Horizon">30</map>
    </bind>
  </component>
</sspaceex>
```