Case Study: System Model of Crane and Embedded Control

Eduard Moser
Robert Bosch GmbH, FV/FLI
Stuttgart, Germany
moser@fli.sh.bosch.de

Wolfgang Nebel
Carl von Ossietzky Universität Oldenburg
and OFFIS
Oldenburg, Germany
nebel@computer.org

Abstract

A case study of a crane is defined for the evaluation of system description languages. The plant (car and load) is given as a fourth-order linear system. The embedded control includes sensors, actuators, two control strategies, and diagnosis.

1. Introduction

An embedded system should be developed for controlling a crane (Figure 1). The crane moves a load along its track to some desired position.

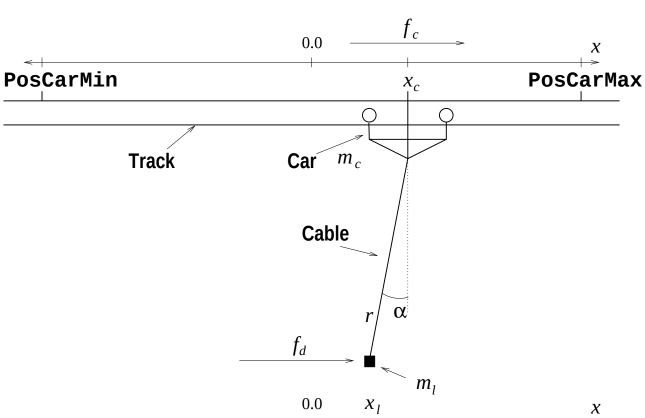

Figure 1. Crane with load

For this case study only a high level description (a model) of the system including the plant and the embedded control should be delivered. The system is given by the specification of the plant (car and load) and the specification of the embedded control.

2. Specification of the plant

The linear description of the crane is given in [1, pp. 357ff]. The position of the car x_c and of the load x_l are given according to the following equations:

$$
\begin{aligned}
\ddot{x}_c &= \frac{f_c}{m_c} + g\frac{m_l}{m_c}\,\alpha - \frac{d_c}{m_c}\,\dot{x}_c \\
r\,\ddot{\alpha} &= -g(1+\frac{m_l}{m_c})\alpha + (\frac{d_c}{m_c} - \frac{d_l}{m_l})\dot{x}_c - r\,\frac{d_l}{m_l}\,\dot{\alpha} \\
&\quad -\frac{f_c}{m_c} + \frac{f_d}{m_l} \\
x_l &= x_c + r\,\alpha
\end{aligned}
$$

The disturbance (e.g., wind) is modelled as the external force f_d accelerating the load. The forces f_c and f_d are the inputs to the system.

For the given system we assume the following constants: $m_c = 10.0$ kg, $m_l = 100.0$ kg, $g = 9.81$ m/sec^2, $d_c = 0.5$/sec, $r = 5.0\,m$, $d_l = 0.01$/sec.

3. Specification of the embedded control

The specification of the embedded control includes the sensors (section 3.1.), the actuators (section 3.2.), and the job control (section 3.3.).

3.1. Sensors (inputs)

The sensors race events. The minimal time between two events caused by one such sensor is 2 ms.

POSCAR position of the car (x_c/[m] with some precision).

SWPOSCARMIN is a range sensor with boolean value, it is true if $x_c <$ POSCARMIN m, with POSCARMIN $= -5.0$.

SwPosCarMax is a range sensor with boolean value, it is true if $x_c > $ PosCarMax m, with PosCarMax $= 5.0$.

Alpha angle ($\alpha/$[rad]) of the cable with some precision. The sensor is constraint to abs(Alpha) $<$ AlphaMax $+ \Delta$Alpha, with AlphaMax $= 0.2$ and ΔAlpha $= 0.01$.

SwShutDown switch which is true if user wants to shut down the system.

PosDesired desired position of load.

3.2. Actuators (outputs)

VC drives a dc-motor which releases the force f_c according to the equations

$$t_m \dot{f}_c + f_c = k_m v$$
$$v = \text{VC volt}$$

which in turn drives the car (with $t_m = 1.0$ sec, $k_m = 4.0$ N/volt). VC is limited to abs(VC) $\leq$ VcMax, with VcMax $= 40.0$.

Brake sets VC $= 0.0$ and switches a relay which controls the brake, if set then the brake is applied on the car (it stops immediately).

EmergencyStop sets VC $= 0.0$ and trips a circuit breaker such that the power is going down. All further activities are blocked (an emergency brake stops the car immediately). The emergency brake is released only if power is reset.

3.3. Job control (running after power on)

After initialisation the job control handles the requested jobs. If the power is switched on the following procedure is executed.

1. Initialisation: Apply regular brake (set Brake). Set $\boldsymbol{q}_0 = [0,\ 0,\ 0,\ 0,\ 0]$ (see section 3.3.3.). Run the sensor check (section 3.3.1.).
2. Unset EmergencyMode. Activate the diagnosis (section 3.3.2.).
3. Loop until SwShutDown:
   (a) Wait for input of next desired position then put value into PosDesired (and hold it).
   (b) Release brake (unset Brake).
   (c) Apply the control algorithm (section 3.3.3.).
4. Stop Activities.

3.3.1. Sensor check. The following procedure is supposed to check the position sensors PosCar, SwPosCarMin, SwPosCarMax. After a first evaluation of the static values the car is driven beyond the minimal position and the values are checked. Then the car is driven beyond the maximal position and the values are checked. Finally, the car is driven into the regular working area. The sensor check assumes a minimal precision of all position sensors as ΔPosCar. ΔPosCar $= 0.1$, VcCheck $= 0.002$. In detail:

1. if (PosCar $<$ PosCarMin $- \Delta$PosCar) $\vee$ (PosCar $>$ PosCarMax $+ \Delta$PosCar) $\vee$ (SwPosCarMin $\wedge$ SwPosCarMax) then set EmergencyStop.
2. Unset Brake and set VC $= -$VcCheck and then proceed until PosCar $\leq$ PosCarMin $- \Delta$PosCar $\vee$ SwPosCarMin. If PosCar $>$ PosCarMin $+ \Delta$PosCar $\vee$ $\neg$SwPosCarMin then set EmergencyStop.
3. Set VC $= +$VcCheck and then proceed until PosCar $\geq$ PosCarMax $+ \Delta$PosCar $\vee$ SwPosCarMax. If PosCar $<$ PosCarMax $- \Delta$PosCar $\vee$ $\neg$SwPosCarMax then set EmergencyStop.
4. Set VC $= -$VcCheck and then proceed until PosCar $\leq 0.0 \vee$ SwPosCarMin. If SwPosCarMin $\vee$ SwPosCarMax then set EmergencyStop otherwise set VC $= 0$ and return from sensor check.

3.3.2. Diagnosis. The diagnosis runs in parallel to the control algorithm after the sensor check is done.

- Check if the desired position is within the allowed interval (with ΔPosSec $= 1.0$): if ($\neg$(PosCarMin $+ \Delta$PosSec $<$ PosDesired $<$ PosCarMax $- \Delta$PosSec)) set EmergencyStop.
- Observe the range sensors: if (SwPosCarMin $\vee$ SwPosCarMax) for more then 20 ms during 100 ms then set EmergencyStop.
- Observe the plausibility of the angle sensor: if (abs(Alpha) $>$ AlphaMax) for more then 50 ms during 100 ms then set EmergencyMode such that the control algorithm works without the angle sensor (extended observer concept).

3.3.3. Control algorithm. The control algorithm is a cycle based algorithm with the fixed cycle time of 10 ms. The index n numbers the time points within the following difference equations. The output VC is given as:

$$
\boldsymbol{A} = \begin{bmatrix} +9.9950e-01, & +9.8040e-01, & +9.9444e-04, & -2.9370e+01, & +4.9032e-03 \\ +4.9882e-07, & +9.9892e-01, & -9.9632e-07, & +4.8569e+01, & +9.9963e-03 \\ +0.0000e+00, & +0.0000e+00, & +9.9004e-01, & -5.2731e+04, & +0.0000e+00 \\ +9.9975e-03, & +4.9033e-03, & +4.9817e-06, & +9.9757e-02, & +1.6346e-05 \\ +9.9739e-05, & -2.1569e-01, & -1.9888e-04, & -9.5679e+01, & +9.9892e-01 \end{bmatrix}
$$

$$
\boldsymbol{B} = \begin{bmatrix} +1.9926e-05, & +2.9370e+01 \\ -1.3296e-08, & -4.8569e+01 \\ +3.9800e-02, & +5.2731e+04 \\ +6.6485e-08, & +9.0024e-01 \\ -3.9853e-06, & +9.5679e+01 \end{bmatrix}
$$

$$
\boldsymbol{K} = [+1.0906e+03, \ -3.2981e+03, \ +5.7106e+00, \ +0.0000e+00, \ +3.0647e+03]
$$

$$
k_p = +5.5000e+02
$$

Figure 2. Parameters for the control algorithm

$$
\begin{aligned}
\boldsymbol{q}_n &= [q_{1n}, q_{2n}, q_{3n}, q_{4n}, q_{5n}]^T \\
\boldsymbol{q}_{n+1} &= \boldsymbol{A} \cdot \boldsymbol{q}_n + \boldsymbol{B} \cdot \begin{bmatrix} \mathrm{VC}_n \\ \mathrm{POSCAR}_n \end{bmatrix} \\
y_n &= \boldsymbol{K} \cdot \boldsymbol{q}_n \\
z_n &= \begin{cases} \mathrm{POSCAR} + (r/\mathrm{m})\, q_{2n} & \text{if EMERGENCYMODE} \\ \mathrm{POSCAR} + (r/\mathrm{m})\ \mathrm{ALPHA} & \text{otherwise} \end{cases} \\
u_n &= k_p\ (\mathrm{POSDESIRED} - z_n) \\
\mathrm{VC}_n &= \begin{cases} +\mathrm{VCMAX} & \text{if } u_n - y_n > +\mathrm{VCMAX} \\ -\mathrm{VCMAX} & \text{if } u_n - y_n < -\mathrm{VCMAX} \\ u_n - y_n & \text{otherwise} \end{cases}
\end{aligned}
$$

The parameters for the algorithm above are given in figure 2.

In parallel an observer detects the breaking condition for the control algorithm: If EMERGENCYMODE and
$abs(\mathrm{VC}) < 0.01$ for 15 sec,
or not EMERGENCYMODE and
$abs(\mathrm{VC}) < 0.01$ for 5 sec,
then apply the brake (set BRAKE) and return.

4. Test cases

For validation of the models the following test suite should be executed. For each of the three cases a transient diagram (x_l and α) and the exact time stamps of all important events (end of Init, each desired position reached, EMERGENCYSTOP, etc.) shall be presented.

The solution method for the differential equations is outside the scope of this specification. As reference, we propose fourth-order Runge-Kutta-Method (step size = 1 ms).

CASE 1:

Init POSCAR $= 0.0$, $f_d = 0.0$ N, set power on.

1. POSDESIRED $= 3.5$.

2. POSDESIRED $= 3.7$.

3. POSDESIRED $= -3.5$. After 4 sec set $f_d = 300.0$ N for 1 sec then $f_d = 0.0$ N again.

4. POSDESIRED $= 3.5$. After 3 sec the angle sensor breaks which sets ALPHA $=$ ALPHAMAX$+0.05$ until the end of the test case.

5. POSDESIRED $= 3.7$.

6. POSDESIRED $= -4.9$.

CASE 2:

Init POSCAR $= 0.0$, $f_d = 0.0$ N, set power on.

1. POSDESIRED $= -3.5$. After 9 sec set $f_d = -300.0$ N for 1 sec then $f_d = 0.0$ N again.

CASE 3:

Init Assume SWPOSCARMAX is defect (= false) permanently, POSCAR $= 0.0$, $f_d = 0.0$ N, then set power on.

Acknowledgements

We would like to thank Roland Karrelmeyer for designing and parametrising the control algorithm (section 3.3.3.).

References

[1] O. Föllinger. *Regelungstechnik*. Hüthig, 5. edition, 1985.