ARGESIM Benchmark C13 ‘Crane and Embedded Control’ with SI MULIN K – modelled Dynamics and MATLAB-programmed Control

Georg Janisch, Miriam Leopoldseder*, Andreas Körner

Institute for Analysis and Scientific Computing, Vienna Univ. of Technology Wiedner Hauptstraße 8
1040 Vienna, Austria; miriam.leopoldseder@tuwien.ac.at

Abstract. ARGESIM Benchmark C13 ‘Crane and Embedded Control’ is based on nonlinear and a linear continuous dynamics for a crane crab and on discrete control by a linear observer model. This solution makes use of the MATLAB/Simulink system in a ‘mixed’ manner. The continuous dynamics are modelled graphically in the Simulink environment – using partly connected blocks and partly MATLAB functions. The discrete control with observer model and the diagnosis is programmed in MATLAB, framed by a sampled date discrete submodel in Simulink connected to the continuous submodels for the dynamics. This approach allows fast modelling, but deeper insight into the simulation environment is necessary.

Introduction

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming [1]. It is based on numerical vector and matrix manipulation. SIMULINK is a graphical block diagram environment for multidomain simulation and model-based design [2]. Within Simulink, MATLAB functions can be used, with high complexity. This solution makes use of the MATLAB/Simulink system in a ‘mixed’ manner.

The continuous dynamics are modelled graphically in the Simulink environment – using partly connected blocks and partly MATLAB functions. The discrete control with observer model and the diagnosis is programmed in MATLAB, framed by a sampled date discrete submodel in Simulink connected to the continuous submodels for the dynamics.

1 Modelling

Basis for the dynamic model are the linear and nonlinear mechanical equations for the crane crab, and the ‘electrical equation for the DC motor

Linear model for crane crab:

\[ \ddot{x}_c = \frac{f_c}{m_c} + g \frac{m_l}{m_c} \alpha - \frac{d_c}{m_c} \dot{x}_c \]  
\[ r \ddot{\alpha} = -g \left(1 + \frac{m_l}{m_c}\right) \alpha + \left(\frac{d_c}{m_c} - \frac{d_l}{m_l}\right) \dot{x}_c - r \frac{d_l}{m_l} \dot{\alpha} \]

Nonlinear model for crane crab:

\[ \ddot{x}_c [m_c + m_l \sin \alpha^2] = -d_c \dot{x}_c + f_c + f_d \sin \alpha^2 + m_l \sin \alpha \left[r \ddot{\alpha} + g \cos \alpha\right] - d_l \dot{x}_c \sin \alpha^2 \]

\[ \ddot{\alpha} \left[m_l \sin \alpha^2 + m_c\right] = \left[f_d \frac{m_c}{m_l} - f_c + \right. \]
\[ d_c \dot{x}_c \left[r \cos \alpha - \left[g (m_l + m_c) + m_l r \alpha \right] \sin \alpha - d_l \left[\frac{m_c}{m_l} (r \cos \alpha + r^2 \dot{\alpha}) + r^2 \dot{\alpha} \sin \alpha^2 \right] \right] \]

\[ x_i = x_c + r \sin \alpha \]

Linear model for DC motor:

\[ t_m \dot{f}_c + f_c = k_m v \]

In general, for dynamic model parts Simulink submodels with continuous blocks were used. For discrete model parts – i.e. the controller and the diagnosis – Simulink submodel(s) with MATLAB functions inside were used (Figure 1 – modelling overview)
The crane dynamics and the DC-motor were built in SIMULINK using continuous blocks with different levels of granularity.

The control is based on a linear observer which operates discrete. With observer matrices $A$, $B$ and $K$ and observer state vector $\bar{q}$, control $v$ is computed in the main by

$$\bar{q}_{n+1} = A\bar{q}_n + B[v_n x_{c,n}]^T, \quad y_n = K\bar{q}_n$$

$$v_{n+1} = k_p(\text{pos}_{\text{desired}} - (x_{c,n} + ra)) - y_{n+1}$$

The controller algorithm and the diagnose functions were programmed in MATLAB and implemented in SIMULINK as a MATLAB function block. These blocks are supposed to work discrete. To assure that Zero-Order-Hold blocks are used. The angle sensor is simulated by a quantizer block and by comparing the current value to maximum and minimum defined in the control function (Figure 1).

The diagnosis is also implemented in form of a MATLAB-function. Fortunately Simulink allows different sample times for discrete blocks, so for the controller function a sample time of 10 ms and for the diagnosis functions a sample time of 1 ms is set.

### 2 Task a: Comparison of Nonlinear and Linear model without Control

The linear and nonlinear models were compared without brake and controller, so that from the model in Figure 1 only the nonlinear submodel and the linear submodel for the crane crab were used, without feedback control, only with given input $f_c$ and disturbances $fd$.

The nonlinear model is an implicit. Before implementation, the model was transformed symbolically into an explicit state space model (with four states for the crane crab, and one state for the motor). As the nonlinearities are relatively comprehensive (after making the model explicit), no basic Simulink blocks were used: the nonlinearities were calculated in a MATLAB function, and fed into a Simulink integrator block.

The dynamic calculations are done by an ODE45-solver which is the standard MATLAB solver for non-stiff ODEs. Below the steady-state difference of $x_l$ and $fd$ disturbance is shown.

<table>
<thead>
<tr>
<th>$fd$ disturbance</th>
<th>-750</th>
<th>-800</th>
<th>-850</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_l$ difference</td>
<td>-5.3086</td>
<td>-6.8225</td>
<td>-7.8131</td>
</tr>
</tbody>
</table>

### 3 Task b: Simulation of the Linear Fully Controlled System

The task was to simulate the fully controlled system in a given scenario. The car starts out at position zero with no external force affecting it and with the desired position $+3$. After 16 seconds the target position is changed to $-0.5$ and then after 36 seconds changed again to $3.8$. At time $t = 42$ $f_d$ is set to $-200$ for 1 second and then to zero. The simulation stops at 60 seconds.

The controller and the brake are implemented via the DC-motor interacting with the linear model (Figure 1) following a linear discrete observer control.

The signal $d_c$ of the model stands for the friction coefficient of the car. To implement the brake, the value of $d_c$ is increased to simulate a mechanical brake. Again, the brake mechanism is described in the MATLAB controller function. When the break is applied the internal control parameters are held constant and the VC output is set to zero. In the linear model the instant stop of the car is achieved by increasing the friction coefficient to $10000$ s$^{-1}$. 

---

**Figure 1.** Overall model with continuous dynamics (linear / nonlinear) and discrete control.
The dynamic calculations are done by an ODE45-solver which is the standard MATLAB solver for non-stiff ODEs, and which also works well with discrete model parts. The simulation results are shown in Figure 2 and Figure 3.

If the range sensors for the maximum or minimum position are activated for more than 20 ms during a 100 ms time period the Emergencystop is initiated. To assure that, every millisecond the states of the range sensors are stored in an array. It has exactly 100 slots and if the sum exceeds 20 the Emergencystop is set.

The Emergencymode is handled the same way, except that it is activated when the position limits are exceeded for more than 50 ms.

The break is applied when the car reaches its maximum position and the Emergencystop is activated. The load oscillates after the stop (results see Figure 4 and Figure 5).

The interaction between the controller and the model is extremely sensible. Minor changes of the circle time have a great impact. They result in a decreased performance of the control.

4 Task c: Simulation of Controlled System with Sensor Diagnosis

In this task the sensor diagnosis was added to the system. During the 60 second simulation time the desired position is shifted from +3 to -0.5. After 18 seconds the angle sensor is forced to trigger the Emergencymode. Then at t = 36 the desired position is changed again to 3.8 after which f_d is set to 200 for 1 s, before resetting it to zero.

The diagnosis is described in a MATLAB block with a sample time of 1 ms. The range sensors are programmed in the diagnosis MATLAB function.

The diagnosis is described in a MATLAB block with a sample time of 1 ms. The range sensors are programmed in the diagnosis MATLAB function.

References

Simulation in Production and Logistics 2015 - 16. ASIM-Fachtagung Simulation in Produktion und Logistik

Simulation in Produktion und Logistik 2013: Entscheidungsunterstützung von der Planung bis zur Steuerung


Simulation and Optimierung in Produktion und Logistik – Praxisorientierter Leitfaden mit Fallbeispielen.

Verifikation und Validierung für die Simulation in Produktion und Logistik - Vorgehensmodelle und Techniken.

Qualitätskriterien für die Simulation in Produktion und Logistik – Planung und Durchführung von Simulationsstudien.
S. Wenzel, M. Weiß, S. Collisi – Böhmer, H. Pitsch, O. Rose (Hrsg.);


M. Bruckner: Agentenbasierte Simulation von Personenströmen mit unterschiedlichen Charakteristiken

S. Emrich: Deployment of Mathematical Simulation Models for Space Management


X. Descovich: Lattice Boltzmann Modeling and Simulation of Incompressible Flows in Distensible Tubes for Applications in Hemodynamics

F. Miksch: Mathematical Modeling for New Insights into Epidemics by Herd Immunity and Serotype Shift

S. Tauböck: Integration of Agent Based Modelling in DEVS for Utilisation Analysis: The MoreSpace Project at TU Vienna;

Ch. Steinbrecher: Ein Beitrag zur prädiktiven Regelung verbrennungsmotorischer Prozesse

O. Hagendorf: Simulation-based Parameter and Structure Optimisation of Discrete Event Systems