Real-Time & Feedback Control

Moris Behnam
Outline

• Introduction
• Feedback control
• Real-time and control systems
• Using control theories in real-time scheduling
Introduction

• Control systems constitute an important subclass of embedded computing systems

• Automotive systems, ECUs are used to implement different feedback control,
  – Engine control, ABS, Active stability control, Cruise control, Climate control, ..

• Avionic
  – Flight control (autopilot, taking-off, landing, direction and altitude control, ..)

• Automation,
  – Robotics, Process control, ..
Control system theory

• A theory that provides means to control the behavior of dynamic systems.

• Feedback control
Feedback control
Feedback control system
Feedback control theory

• Given a physical system, it is required to control the behavior of the system according to certain control requirements.
• The control performance matrices can be:
  • Stability of the system
  • Maximum overshoot
  • Steady state error
  • Rising time
  • Phase and gain margin
Feedback control theory

Unstable system

Stable response time

Controller

Planet

Sensor
Feedback control design

Physical system → Modeling → Dynamic model → Controller design → Controller realization → Controller model

Controller requirements
System modeling

- Most of control systems are **dynamic** in nature so differential equations can only describe their dynamic behavior.

\[ a_2 \ddot{y}(t) + a_1 \dot{y}(t) + a_0 y(t) = b_1 r(t) + b_0 r(t) \]
System modeling

- Transfer function (Laplace transform) linear time invariant

\[ P(s) = \frac{O(s)}{I(s)} = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} \]

- Block-diagram

- Linear time invariant system, relationship between \( O(t) \) and \( I(t) \) is linear and all \( a_i \) and \( b_i \) are constant as a function of time.
Controller design

- Different methods to design controllers (PID, pole placement, state-feedback, LGQ, ....)
- PID (Proportional integral derivative controller)

\[
\frac{c(s)}{e(s)} = K_p + K_I \frac{1}{s} + K_D s
\]

\[
G(s) = \frac{Y(s)}{U(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0}
\]
Controller design

- Simple pant
  \[ p(s) = \frac{a}{bs + c} \]

- Applying PID, the transfer function is
  \[ G(s) = \frac{k_d s^2 + ak_p s + k_i}{(k_d + b)s^2 + (k_p + c)s + k_i} \]

- Stability of the system
- Maximum overshoot
- Steady state error
- Rising time
- Phase and gain margin
Controller realization

- **Computers control**: Most of the control systems are implemented nowadays based on computer control systems.
Computer control

- Time becomes discrete
- Z-transform is used to convert a discrete time representation to frequency domain
- Transfer function (Z-transform)

\[
G(z) = \frac{Y(z)}{U(z)} = \frac{b_1z + b_0}{a_2z^2 + a_1z + a_0}
\]

- Z-transform can be converted to difference equations

\[
Y(n) = a_1Y(n - 1) + a_2Y(n - 2) + b_1U(n - 1) + b_2U(n - 2)
\]

- Simple to be implemented
Computer Control

- The **Nyquist–Shannon sampling theorem**: sampling frequency = 2 * highest signal frequency.
- In control, sampling frequency = 5-20 * highest frequency.
Computer Control

• Implementation control task
  – Is executed **periodically**
  – $T_c_i \times \omega_c_i = 0.05 \sim 0.2$

//Control task $T_c_i$
t = CurrentTime;
LOOP
  AD-Conversion;
  ControlAlgorithm;
  DA-Conversion;
  t = t + T_c_i;
WaitUntil(t);
END
Real-time & control

- Control task:
  - Periodic task
  - Hard real time task, i.e., it must meet its deadline.
  - Because of OS scheduling, control tasks may suffer from jitter in the sampling and actuation. The problem exists for both offline and online scheduling.
Real-time & control

- Distributed systems: another source of jitter in sampling and actuation comes from the communication delay.
Real-time & control

• The jitter in the input (sampling) and output (actuation) of the control task is modeled as a time varying delay in the input and output

• Time delay may **degrade** the control system performance

• The control engineers do not consider the jitter from the **implementation** in the controller design

• The computer engineers do not know the effect of jitter on the control **performance**
Real-time & control

- Example:

\[
G(s) = \frac{1000}{s(0.5s + 1)}
\]

<table>
<thead>
<tr>
<th>Task</th>
<th>Ti</th>
<th>Ci</th>
</tr>
</thead>
<tbody>
<tr>
<td>task_{PID}</td>
<td>2 ms</td>
<td>0.2 ms</td>
</tr>
<tr>
<td>task_{1}</td>
<td>1.3 ms</td>
<td>0.5 ms</td>
</tr>
</tbody>
</table>
Real-time & control

- The effect of jitter on control loop is decreased when **decreasing the period** of the control task (Controller design is based on the control task period)
- Decreasing the period of control task **increases the task utilization** which decreases the schedulability of the task set.
- It is required to **decrease or eliminate** the effect of jitter on control tasks while guaranteeing the schedulability of the hard real time tasks (including control tasks).
- There are two proposed solutions
  - Decreasing the jitter on control tasks
  - Compensating the effect of jitter on control tasks
Jitter control methods

• Comparative Assessment and Evaluation of Jitter Control Methods, Giorgio Buttazzo, Anton Cervin, RTNS 2007
Task splitting

A- Task splitting (RJTS):

```plaintext
// Control task Tc_i
t = CurrentTime;
LOOP
    AD-Conversion;
    ControlAlgorithm;
    DA-Conversion;
    t = t + T;
    WaitUntil(t);
END
```
Task splitting

• **Advantages**
  – Decreases the jitter in input and output significantly.
  – Constant delay can be handled by the controller design.
  – Can be applied to all types of control task.

• **Disadvantages**
  – Imposes longer delay.
  – Implementation overhead and complexity
  – Decreases schedulability

• The control algorithm part of control tasks can be further splitted into tow parts, controller computation subpart and state updating part.
Advancing deadlines

B- Advancing deadlines RJAD:
  – Shorten the control tasks relative deadlines to decrease the jitter.
Advancing deadlines

- **Advantages**
  - Simple to be implemented without requiring special support from OS
  - Can decrease the effect of the jitter
  - Does not impose extra runtime overhead

- **Disadvantages**
  - Can not reduce the jitter for all control tasks
  - Decreases the schedulability of the system
Non-preemption

C- Non-preemption RJNP
- Control tasks are executed non-preemptively
Non-preemption

• Advantages
  – Decreases the delays and jitters

• Problems
  – Decreases the schedulability
  – No utilization upper bound can be used
  – Certain OS do not support non-preemptive execution of tasks.
Jitter control methods

• The three approaches evaluation:
  – Input Jitter INJ, Input-Output Jitter IOJ, Input-Output latency IOL
Jitter control methods
Jitter compensation

• Jitter Compensation for Real-Time Control Systems, Pau Martí and Josep M. Fuertes, Gerhard Fohler, Krithi Ramamritham.
Jitter compensation

- Jitter compensation:
  - It adjusts the controller parameters during runtime to compensate the effect of jitter in the sampling and actuation.
  - It either measures the jitter and delay in the sampling and/or actuation online or it computes it from offline knowing the scheduler and the task set. The execution time of the control task should be provided or estimated.
  - The controller equation is expressed as a function of the jitter in sampling and actuation:

\[
\begin{align*}
  x(\bar{h}_{k+1}) &= \Phi(h_k)x(\bar{h}_k) + \Gamma_0(h_k, \tau_k)u(\bar{h}_k) + \Gamma_1(h_k, \tau_k)u(\bar{h}_{k-1}) \\
  y(\bar{h}_k) &= Cx(\bar{h}_k) + Du(\bar{h}_k) \\
  u(\bar{h}_k) &= -L(h_k, \tau_k)x(\bar{h}_k) \\
  \bar{h}_k &= \sum_{0}^{k} h_k
\end{align*}
\]
Jitter compensation

• Advantages
  – Eliminates the effect of jitter on the control performance.
  – Takes into account the effect of the real time scheduling in the controller design.

• Disadvantages:
  – Since it is online method, it adds runtime overhead specially if it measures the jitter online.
  – If the measurements are done offline based on the scheduling algorithm, it will require a memory space to save the setting for each cycle. Also it might not be accurate as the execution time of tasks may vary.
Jitter compensation

- PID controller
Jitter compensation

- Example:

<table>
<thead>
<tr>
<th></th>
<th>Task1</th>
<th>Task2</th>
<th>Control task</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>60ms</td>
<td>70ms</td>
<td>80ms</td>
</tr>
<tr>
<td>C</td>
<td>10ms</td>
<td>10ms</td>
<td>1ms</td>
</tr>
</tbody>
</table>

![System response graphs](image)
Jitter margin

Jitter margin

- Controller requirements: using jitter margin

![Graph 1: Worst-case performance degradation bound for DC servo controlled by PID controller.](image1)

![Graph 2: Jitter margin for LQG controller, designed for delay=5ms](image2)
Jitter margin

• Using offline scheduling to enforce the minimum delay and the maximum allowed jitter.
• Use simulated annealing algorithm to find a valid solution (schedule)
• Advantages
  – Uses the controller requirements to design the scheduler
  – Simple to be implement if the OS supports offline scheduling
  – Does not impose runtime overhead and special support from OS
• Disadvantages
  – No guarantee to find a solution (simulated annealing)
  – If the LCM of control tasks is high it will require long time for the algorithm and large memory to save the schedule. (Since a range of periods is given for each controller, a careful selection of periods can decrease this problem)
Jitter margin

• Example:

\[ P_1(s) = \frac{8 \times 10^5}{s^2 + 1000s}, \quad C_1(s) = \frac{6857s + 8 \times 10^6}{s^2 + 4736s + 1.071 \times 10^7} \]

\[ P_2(s) = \frac{4 \times 10^4}{s^2 - 4 \times 10^4}, \quad C_2(s) = \frac{1.796 \times 10^5s + 4.811 \times 10^7}{s^2 + 5331s + 1.417 \times 10^7} \]

\[ P_3(s) = \frac{5 \times 10^7}{s^3 + 100s^2 + 2.5 \times 10^5s}, \quad C_3(s) = \frac{2.385 \times 10^5s^2 + 4.032 \times 10^7s + 3.788 \times 10^{10}}{s^3 + 10^4s^2 + 3.316 \times 10^7s + 4.059 \times 10^{10}} \]

### Controller requirements

<table>
<thead>
<tr>
<th>System</th>
<th>( \phi_m )</th>
<th>( h_{max} ) (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1(s), C_1(s) )</td>
<td>35°</td>
<td>1.00</td>
</tr>
<tr>
<td>( P_2(s), C_2(s) )</td>
<td>30°</td>
<td>1.50</td>
</tr>
<tr>
<td>( P_3(s), C_3(s) )</td>
<td>30°</td>
<td>3.00</td>
</tr>
</tbody>
</table>

### Proposed solution

<table>
<thead>
<tr>
<th>System</th>
<th>( h )</th>
<th>Min. delay</th>
<th>Deadline</th>
<th>Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1(s), C_1(s) )</td>
<td>0.45</td>
<td>0.29</td>
<td>0.50</td>
<td>0.01</td>
</tr>
<tr>
<td>( P_2(s), C_2(s) )</td>
<td>0.50</td>
<td>0.20</td>
<td>0.22</td>
<td>0.02</td>
</tr>
<tr>
<td>( P_3(s), C_3(s) )</td>
<td>0.60</td>
<td>0.45</td>
<td>0.46</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### RM

<table>
<thead>
<tr>
<th>System</th>
<th>( h )</th>
<th>Jitter</th>
<th>( \phi_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1(s), C_1(s) )</td>
<td>0.35</td>
<td>0</td>
<td>65°</td>
</tr>
<tr>
<td>( P_2(s), C_2(s) )</td>
<td>0.50</td>
<td>0.15</td>
<td>30°</td>
</tr>
<tr>
<td>( P_3(s), C_3(s) )</td>
<td>1.50</td>
<td>0.30</td>
<td>-10°</td>
</tr>
</tbody>
</table>
Network control system

• Network control system
  – Network scheduling is similar to OS scheduling
  – Some of the techniques used to decrease the effect of the input and input-output jitter in OS can be used also in the network scheduling.
  – Packet transmission is non-preemptive and Control task is already split into three subtasks located in different nodes.
  – Changing the priority of messages may not be as easy as changing the priority of tasks.
  – Time synchronization (global time) should be supported if it is required to use the solution based on jitter compensation.
  – The source of the jitter/delay is from the network scheduling and blocking

• Many of the solutions proposed to NCS are based on the fact that the urgency of the control task becomes less when the system response is in the steady state.
Network control system

Network control system

- CAN bus
  - Widely used in industry
  - Fixed priority scheduling
  - Message Priority is specified in the identifier field (ID field) of the frames
  - The scheduling is done by comparing bit by bit of the ID field starting from the Most Significant Bit (MSB).
  - In CAN the bit 0 is a dominant bit and the bit 1 is a recessive bit.
Network control system

- To decrease the jitter on control tasks, their priorities should be the highest in the network, however, it might not be possible to do this due to having many control loops using the network.
- One solution can be though increasing the priority of the frames when it is required (during the transient state).

![Diagram of network control system](image)

- First level is the class priority.
- Second level is the urgency priority.
Network control system

• Let $u(k)$ be the controller output at sample $k$, the priority of frames will be proportional to $u(k)$

• The computation of the priority is done in the controller node

• The sensor node will also receive the control action when it is sent to the actuator node, so it adapts the priority of the sensor frame
Network control system

Static priority

Dynamic priority f1

Dynamic priority f3
Feedback scheduling

• Towards Adaptive Hierarchical Scheduling of Real-time, Nima Moghaddami Khalilzad, Thomas Nolte, Moris Behnam, Mikael Åsberg, ETFA 2011
Resource reservation
Static Portions of CPU (Budgets)

Static systems
- Execution times are fixed
- Tasks are not added or removed during run-time
Feedback scheduling

- Execution times are changing during run-time.
- Tasks are added or removed during run-time.
System Modeling

**Controlled variables:**
1. Idle time in each subsystem
2. Number of deadline missed tasks

**Manipulated variable:**
1. Budget of subsystems
Controller design

**Controller**
- PI controller for each feedback loop
- Mathematical model of the control system is evaluated
- Stability analysis is used to evaluate the controller parameters
- Manual tuning
Idle time feedback loop

\[ P = K_p \Delta(t) \]

\[ I = K_i \int \Delta(t) \, dt \]

Subsystem: New Budget + U-loop error + Reference Input (U_{Set})
Simulation Results
## Simulation Results

<table>
<thead>
<tr>
<th>Subsystems</th>
<th>Budget</th>
<th>3</th>
<th>2</th>
<th>adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline Misses</td>
<td></td>
<td>12</td>
<td>33</td>
<td>4</td>
</tr>
<tr>
<td>Idle time</td>
<td></td>
<td>197</td>
<td>77</td>
<td>185</td>
</tr>
</tbody>
</table>
Control & real-time

• Other research directions
  – Mode change: for example changing the sampling period of controllers requires to change the parameters of the controller online and that may make the control system unstable
  – Best effort: many controller models will be available with different execution time requirements and selecting the best one at each time depends on the available resources
Assignment

• 4 different control systems ($C_1, C_2, C_3, C_4$), $C_1$ and $C_2$ are located in the same node while $C_3, C_4$ are distributed and their controller calculation tasks share the same node with $C_1$ and $C_2$. OS scheduling FPS, network CAN bus, there are more important frames that have higher priority than the sampling and actuation messages ($f_1, ..., f_j$) and each has a period and transmission time ($T_i, t_{tr_i}$). Assuming that a special time is used to read the sensor input (sampling jitter is not a problem) and given an acceptable range of jitter in the actuation for each control loop ($J_1, J_2, J_3, J_4$), propose an approach (equations) that guarantee the schedulability of the tasks and limits at the same time the jitter in the actuation. You should also include all possible runtime overhead (OS and communication).