Real-time for control systems

March 7, 2017

G. De Tommasi[1]
[1]DIETI, Università di Napoli Federico II

Outline

Real-time systems

Real-time for control

What is a real-time system?

- A real-time system is a system (hardware+software) subject to “real-time constraints”.

What is a real-time system?

- A real-time system is a system (hardware+software) subject to “real-time constraints”.
- In a real-time system, the result of a computation is correct if
  - is correct (!)...

What is a real-time system?

- A real-time system is a system (hardware+software) subject to "real-time constraints".
- In a real-time system, the result of a computation is correct if
  - is correct (!)...
  - ... AND meets specified time constraints – the so called "deadlines"

Non-real-time VS real-time - 1/2

Example of non-real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`

Non-real-time VS real-time - 1/2

Example of non-real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`

```
double weightedSum(double u1, double u2, double w1, double w2){
            double result = w1*u1+w2*u2;
            return result;
}
```

Non-real-time VS real-time - 2/2

Example of Real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`

Non-real-time VS real-time - 2/2

Example of Real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`
- **Non-functional requirement:** perform the computation in **at most** 1 ms

Non-real-time VS real-time - 2/2

Example of Real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`
- **Non-functional requirement:** perform the computation in **at most** 1 ms

Now writing. . .

```
double weightedSum(double u1, double u2, double w1, double w2){
              double result = w1*u1+w2*u2;
              return result;
}
```

. . .**is no more sufficient to fulfill the requirements!**

Example of Real-time algorithm

- **Functional requirement:** Given the two weights w1 and w2, compute the weighted sum of the two inputs u1 and u2
- **Non-functional requirement:** perform the computation in **at most** 1 ms

Now writing...

```
double weightedSum(double u1, double u2, double w1, double w2){
             double result = w1*u1+w2*u2;
             return result;
}
```

...is no more sufficient to fulfill the requirements!

We should exploit (indirectly) the hardware architecture and (directly) the operating system, in order to meet the time constraint

Time constraints 1/2

- A computation must be performed **every X time units**

Time constraints 1/2

- A computation must be performed **every X time units**
  - is a *periodic* activity (task), and the time constraint must be met with a given accuracy (*jitter*)

Time constraints 1/2

- A computation must be performed **every X time units**
  - is a *periodic* activity (task), and the time constraint must be met with a given accuracy (*jitter*)

Examples

- "the control action to be applied by the aerosurfaces of an aircraft must be computed every 5 ms"

Time constraints 1/2

- A computation must be performed **every X time units**
  - is a *periodic* activity (task), and the time constraint must be met with a given accuracy (*jitter*)

Examples

- "the control action to be applied by the aerosurfaces of an aircraft must be computed every 5 ms"
- "System A must send a message to system B every 10 s"

Time constraints 1/2

- A computation must be performed **every X time units**
  - is a *periodic* activity (task), and the time constraint must be met with a given accuracy (*jitter*)

Examples

- "the control action to be applied by the aerosurfaces of an aircraft must be computed every 5 ms"
- "System A must send a message to system B every 10 s"
  - **Remember: real-time does not necessarily means "fast"!**

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**
  - is a task with a *deadline* (*cyclic* or *event-based*)

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**
  - is a task with a *deadline* (*cyclic* or *event-based*)

Examples

- "the cyclic execution of a PLC must terminates within 200 ms"

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**
  - is a task with a *deadline* (*cyclic* or *event-based*)

Examples

- "the cyclic execution of a PLC must terminates within 200 ms"
- "stop the cruise control within 50 ms after the break press"

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**
  - is a task with a *deadline* (*cyclic* or *event-based*)

Examples

- "the cyclic execution of a PLC must terminates within 200 ms"
- "stop the cruise control within 50 ms after the break press"

- **Note: usually a periodic task should also meet a deadline**

Hard and soft real-time

- **Hard** real-time systems
  - **Missing (even a single) deadlines means system failure (!)**

Hard and soft real-time

- **Hard** real-time systems
  - **Missing (even a single) deadlines means system failure (!)**
- **Safety critical** systems
  - Missing deadlines can cause serious loss

Hard and soft real-time

- **Hard** real-time systems
  - **Missing (even a single) deadlines means system failure (!)**
- **Safety critical** systems
  - Missing deadlines can cause serious loss
- **Soft** real-time systems
  - Deadlines may be missed and mainly cause a deterioration of the QoS

Hard and soft real-time

- **Hard** real-time systems
  - **Missing (even a single) deadlines means system failure (!)**
- **Safety critical** systems
  - Missing deadlines can cause serious loss
- **Soft** real-time systems
  - Deadlines may be missed and mainly cause a deterioration of the QoS

- Real world (real-time) systems have a mix of hard/soft components
- The distinction between hard and soft real-time is somewhat subjective
- **Soft real-time is not Non-real-time (!)**

Assess schedulability

- Given n real-time tasks. . .

Design real-time systems

Assess schedulability

- Given n real-time tasks. . .
- . . .given the correspondent time constraints (deadlines). . .

Design real-time systems

Assess schedulability

- Given n real-time tasks. . .
- . . .given the correspondent time constraints (deadlines). . .
- . . .given the hardware (and software) architecture. . .

Design real-time systems

Assess schedulability

- Given n real-time tasks...
- ...given the correspondent time constraints (deadlines)...
- ...given the hardware (and software) architecture...
  - is it possible to meet all the timing requirements, i.e. is it possible to schedule the tasks?

Assess schedulability

- Given n real-time tasks...
- ...given the correspondent time constraints (deadlines)...
- ...given the hardware (and software) architecture...
  - is it possible to meet all the timing requirements, i.e. is it possible to schedule the tasks?
    - Are the deadlines met for all the cyclic and event-driven tasks?

      `End_time(task_k)-Start_time(task_k)` $\leq$ `Deadline(task_n)`

Assess schedulability

- Given n real-time tasks...
- ...given the correspondent time constraints (deadlines)...
- ...given the hardware (and software) architecture...
  - is it possible to meet all the timing requirements, i.e. is it possible to schedule the tasks?
    - Are the deadlines met for all the cyclic and event-driven tasks?

      `End_time(task_k)-Start_time(task_k)` $\leq$ `Deadline(task_n)`
    - Are the periodic tasks executed with the required accuracy? Do they meet their deadlines?

Assess schedulability

- Given n real-time tasks...
- ...given the correspondent time constraints (deadlines)...
- ...given the hardware (and software) architecture...
  - is it possible to meet all the timing requirements, i.e. is it possible to schedule the tasks?
    - Are the deadlines met for all the cyclic and event-driven tasks?

      `End_time(task_k)-Start_time(task_k)` $\leq$ `Deadline(task_n)`
    - Are the periodic tasks executed with the required accuracy? Do they meet their deadlines?
- There exist formal methods that permits to assess schedulability (under given assumptions)

Real-time operating systems (RTOS)

- Interrupts/Polling
- Multitasking (concurrency)
- Timer support
- Static scheduling/Preemptive scheduling (priorities)
- Task Segregation
- ...

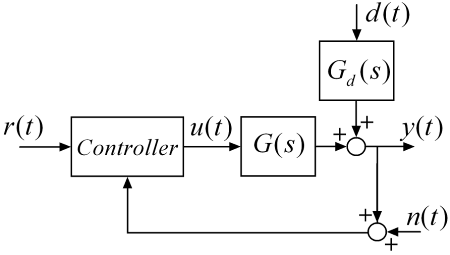
Real-time operating systems (RTOS)

- Interrupts/Polling
- Multitasking (concurrency)
- Timer support
- Static scheduling/Preemptive scheduling (priorities)
- Task Segregation
- ...

Some RTOS

- WindRiver VxWorks
- QNX Neutrino
- RTAI (Linux patch)
- FreeRTOS
- Windows CE

Example - Continuous control system 1/2



The plant

$$G(s) = \frac{2.5 \cdot 10^5}{(s+10)(s^2+80s+2500)}$$

Example - Continuous control system 2/2

The continuous-time controller

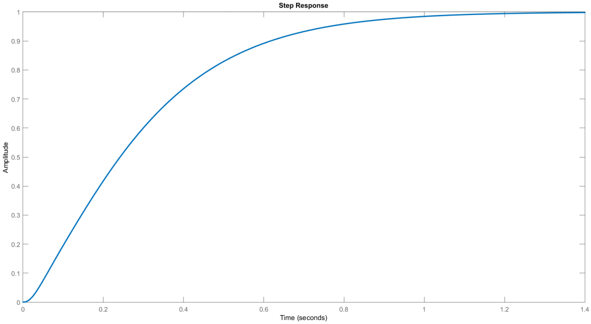
$$C(s) = \frac{2.24(s+25)^2}{s(s+200)} \quad (1)$$

Example - Continuous control system 2/2

The continuous-time controller

$$C(s) = \frac{2.24(s+25)^2}{s(s+200)} \qquad (1)$$

Open-loop step response



Example - Controller discretization

The discrete-time controller

Given the sampling frequency $f_s = 200$ Hz, the Tustin approximation of the controller (1) is

$$\hat{C}_d(z) = \frac{1.686(z-0.882)^2}{(z-1)(z-1/3)} \tag{2}$$

Example - Controller discretization

The discrete-time controller

Given the sampling frequency $f_s = 200$ Hz, the Tustin approximation of the controller (1) is

$$\hat{C}_d(z) = \frac{1.686(z - 0.882)^2}{(z-1)(z-1/3)} \tag{2}$$

- Implementing the discrete-control law (2) means
  - **Functional requirement:** to write a task that computes the correspondent difference equation

Example - Controller discretization

The discrete-time controller

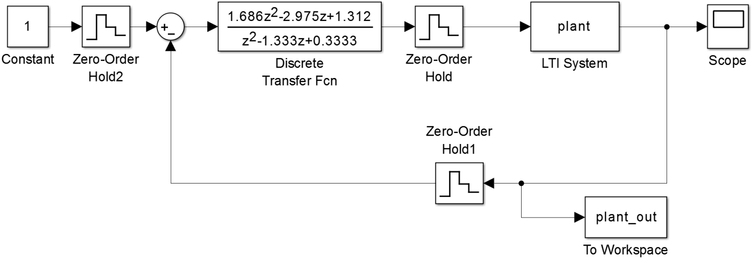
Given the sampling frequency $f_s = 200$ Hz, the Tustin approximation of the controller (1) is

$$\hat{C}_d(z) = \frac{1.686(z - 0.882)^2}{(z-1)(z-1/3)} \tag{2}$$

- Implementing the discrete-control law (2) means
  - **Functional requirement:** to write a task that computes the correspondent difference equation
  - **Non-functional requirement:** to execute the task every 5 ms (assuming negligible execution time)

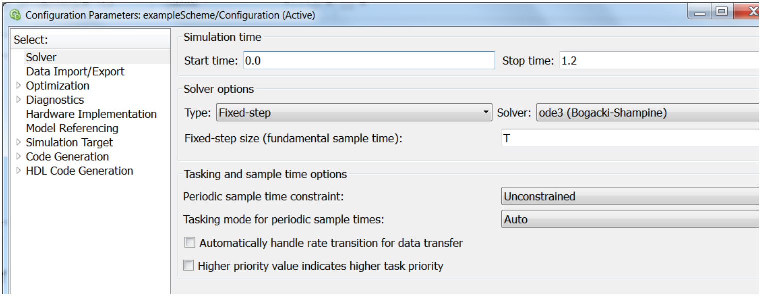
To meet or not to meet (the deadlines)?

Use Simulink. . .



To meet or not to meet (the deadlines)?

. . .with Fixed time-step solver. . .



To meet or not to meet (the deadlines)?

. . .changing the time step