Seminar 3

Summary

- Use of mCs and DSPs in signal processing and control applications
- FIR and IIR filters
- PI and PID regulators
- Predictive regulators (basics)

Seminar 3

Basic references

1. D. Glover, J.R. Deller, "Digital Signal Processing and the Microcontroller", Prentice Hall, 1999.
2. A.V. Oppenheim, R.W. Schafer, J.R. Buck, "Discrete Time Signal Processing", Second Edition, Prentice Hall.
3. K. Ogata, "Discrete Time Control Systems", Prentice Hall, 1987.
4. M. Morari, E. Zafirou, "Robust Process Control", Prentice Hall, 1989.

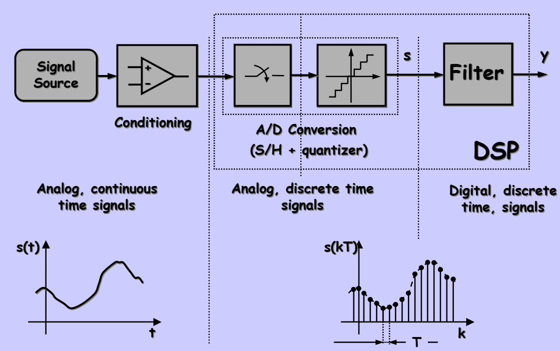
Digital Signal Processing

One of the more frequent uses of mCs and DSPs is in digital signal processing applications and/or real-time control of processes and systems.

The fundamental difference between the two is represented by feedback, not present in the first case, fundamental in the second one.

The problems encountered in these applications are related to discrete time operation of processors, to the finite precision of their arithmetic unit and to the quantization of data and coefficients.

Digital Filters



Digital Filters

In a digital filter, a signal is acquired by the mC or DSP through a A/D converter.

This process implies two effects: sampling and quantization.

Sampling changes the signal from continuous time s(t) to discrete time s(kT).

Quantization changes the signal from analog to digital.

The elaboration takes places on a sequence of quantized samples and generates a new sequence (y).

Digital Filters

As with analog filters, digital filters may have different characteristics:

1. low pass;
2. high pass;
3. band pass or notch;

depending on their frequency response behavior.

Any analog filter can be turned into a digital equivalent within a given precision. Vice-versa is not true: some digital filters do not have analog equivalents (FIR filters).

Digital Filters

Any digital filter can be written as a n-th order difference equation such as:

$$y(k) = b_0x(k) + b_1x(k-1) + \dots + b_nx(k-n) +$$
$$+ a_1y(k-1) + a_2y(k-2) + \dots + a_my(k-m)$$

When one, at least, of the a_i coefficients is <> 0 the filter is called IIR (infinite impulse response).

A FIR filter (finite impulse response) is instead characterized by having all the a_i coefficients equal to zero.

IIR digital low pass filter

A very simple example of a IIR digital low pass filter is given by:

$$y(k) = b\cdot x(k) + (1-b)\cdot y(k-1)$$

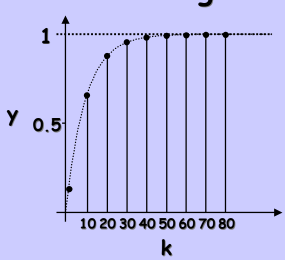
It is easy to analyze the filter's step response, i.e. its response to the input sequence g = {1, 1, 1, 1, 1, … }.

Choosing, for instance, b = 0.1 we get:

y = {0.1, 0.19, 0.27, 0.34, 0.41, … }.

The sequence y goes to 1, but with a infinite duration transient. This is due to the term y(k-1). We also call this a recursive filter.

IIR digital low pass filter



Our IIR filter responds to a step input as if it was the sampled version of a first order low-pass analog filter.

Its speed of response, with respect to the sampling period, depends on our choice of *b*. The bigger *b*, but < 1 (!), the faster the speed of response.

IIR digital low pass filter

Varying parameter b between 0 and 1 we can approximate any first order low pass analog filter. The x(k) and y(k-1) coefficients could be chosen freely, but:

1. if the sum of the coefficients is equal to 1 then the dc gain of the filter is unity;
2. if the y(k-1) coefficient has magnitude < 1 then the filter is also stable.

As an example, the filter:

$$y(k) = 2.1\cdot x(k) - 1.1\cdot y(k-1)$$

is unstable!

IIR digital low pass filter

When we need to get a very fast speed of response, we may want to choose values for b very close to 1.

In this case, however, the finite precision of the processor we are using limits our capability to represent the coefficients!

For instance, in an 8 bit processor, if we choose b > 0.992 we are no longer in a condition to correctly represent 1-b.

Indeed, the minimum number we will be able to represent will be $2^{-7} \cong 0.008$. Lower numbers are all "seen" as 0.

FIR digital low pass filter

We can get a similar low pass filter also without using recursion. For example, a filter like the following:

$$y(k) = \frac{1}{N}\cdot\sum_{i=0}^{N-1} x(k-i)$$

is called N-th order moving average filter. It is basically a low pass filter, but its response gets to the steady state after N sampling periods. As always with FIR filters, there is no stability problem, even in case of a wrong coefficient choice.

FIR digital low pass filter

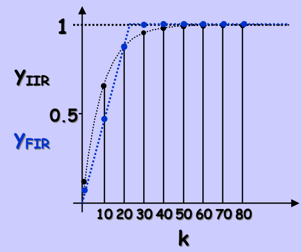
Considering, as an example, N=4, the filter step response (g = {1, 1, 1, … }) is given by sequence y = {0.25, 0.5, 0.75, 1, 1, 1, … }.

Thus, after only 4 steps the transient is over. A similar response cannot be achieved from any analog filter.

We may observe that, to make the two filter responses similar to each other, we need to take a much higher order for the FIR filter (or a much higher b value for the IIR filter).

FIR digital low pass filter

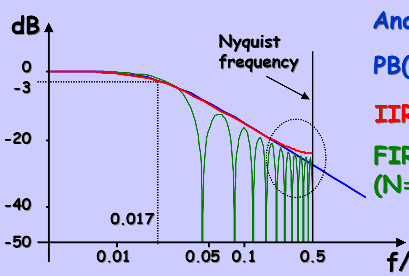
Considering, for instance, N=22, the two filters respond in a similar way.



Thus, the FIR filter requires a bigger number of operations to give a step response similar to the IIR filter's one (22 terms instead of 2). This always happens with FIR filters.

Frequency Response

Digital filters can be described also by means of their frequency response.



Analog filter:

$$PB(s) = \frac{1}{9s + 1}$$

IIR Filter

FIR Filter (N=22)

Frequency Response

Looking at the three frequency responses (taking into account only its magnitude) we see that the filters have a similar behavior.

The FIR filter exhibits frequency cancellation phenomena, due to the periodicity of its structure. The envelope of its frequency response magnitude, follows that of the IIR filter and of the reference analog filter.

The IIR filter and the analog one respond in practically identical ways (the digital filter is indeed the discretization of the analog one).

IIR digital high pass filter

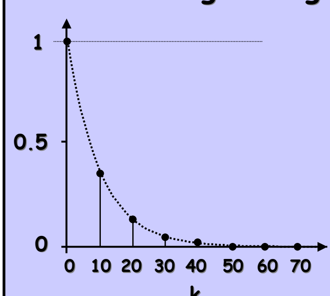
A simple high pass filter can be obtained using the following difference equation:

$$y(k) = x(k) - x(k-1) + a \cdot y(k-1)$$

Being a recursive equation, it corresponds to a IIR filter. Parameter *a* allows to tune the filter response.

Everything is OK if $0 < a < 1$, otherwise we may get (damped) oscillatory step responses or even unstable ones. Actually, it is better to take, at least, $a > 0.5$.

IIR digital high pass filter



Our IIR high pass filter responds to a step input as if it was the sampled version of a first order high pass analog filter.

The graph is obtained with $a = 0.91$. Lower *a* values produce faster responses.

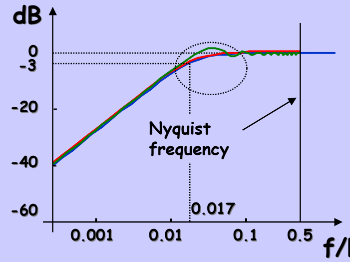
FIR digital high pass filter

Again, we can get a similar filter without using recursion. The following filter:

$$y(k) = \frac{N-1}{N} \cdot x(k) - \frac{1}{N} \cdot \sum_{i=1}^{N-1} x(k-i)$$

is a N-order FIR high pass filter. Its step response reaches the steady-state after N sampling periods. To get a similar response with respect to the IIR filter, we need to take relatively high N values (>20).

Frequency Response



Analog filter

$$PB(s) = \frac{10s}{10s + 1}$$

IIR Filter

FIR Filter (N=22)

In this case we get again very similar frequency responses: note that the filters have relatively low band pass frequencies.

Discretization

We can always use discretization techniques to turn a continuous time filter into an equivalent discrete time one.

The easiest way to do this is using a suitable approximation of the integral operator (1/s) in the discrete time domain, as for example that based on the Euler approximation:

$$\int_0^{nT} x \, dt \cong \sum_{k=1}^{n} x(k) \cdot T$$

where T is the so called integration step.

Discretization

We can then write:

$$Int\_x(nT) \cong \underbrace{T \cdot [x(1)+x(2)+...+x(n-1)}_{Int\_x[(n-1)T]}+x(n)]$$

that is

$$Int\_x(nT) = Int\_x[(n-1)T] + T \cdot x(n)$$

From this we get:

$$\frac{1}{s} = \frac{T}{1-z^{-1}} \Rightarrow s = \frac{1-z^{-1}}{T}$$

Unity delay (z^{-1})

Discretization

Because this is an approximation process, the discretization does not maintain the filter frequency response unaltered. Indeed the original filter frequency response is perturbed and warping phenomena appear. We therefore need to be very careful when applying this method, to avoid *unexpected* digital filter behaviors. As a rule of thumb, we say that discretization results are accurate only up to frequencies equal to 1/10 of the sampling frequency.

Discretization

We may use even more sophisticated discretization methods, that allow us to obtain a better frequency response approximation. For instance:

$$s = \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}}$$

Trapezoidal integration, or Tustin transform.

Finally, we have several methods based on some kind of response invariance to a particular family of signals, like steps or ramps.

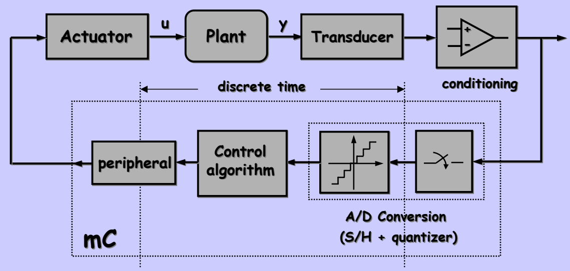
Real-time control

Closed loop digital control of systems or processes, requires the mC or DSP to elaborate signals taken from the plant, according to suitable algorithms, implementing different kinds of regulators.

The design of such regulators requires the application of discrete time automatic control theory.

Their implementation is done using the same signal processing techniques we described for digital filter synthesis.

Real-time control



Closed loop control system with mC

Real-time control

The synthesis of regulators can be done again following different strategies.

The simplest one consists in the discretization of regulators that have been previuosly designed in the continuous time domain.

In most cases, these are just simple PID regulators.

As an alternative, it is possible to use control algorithms that have no continuous time domain equivalent, such as, for instance, the various types of predictive controllers.

Real-time control

It is worth noting that, in most cases, the regulators adopted in industrial applications are just PID controllers.

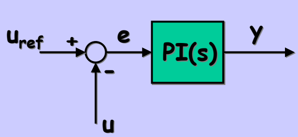
PID controllers usually represent a very good trade-off between complexity and achievable performance.

They are extremely robust and relatively easy to tune (small number of parameters).

Achievable performance is often more than satisfactory, even if always lower with respect to that offered by their analog counterparts.

PI Regulator

In the analog domain, a PI regulator is described by an input-output relation of the following type:



$$\frac{Y(s)}{E(s)} = k_p + \frac{k_i}{s}$$

k_p = proportional gain

k_i = integral gain

PI Regulator

By direct discretization, it is possible to turn the continuous time PI regulator into a suitable control algorithm. Of course, k_p and k_i constants ought to be known already!

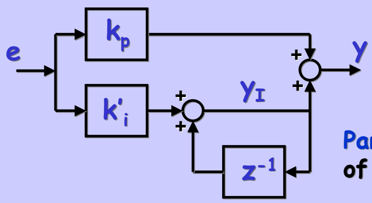
We then immediately find the following control equations:

$$\begin{cases} y(k) = k_p \cdot e(k) + y_I(k) \\ y_I(k) = k_i \cdot T \cdot e(k) + y_I(k-1) \end{cases}$$

$y_I(k)$ ← integral control

$k_i \cdot T$ ← k_i'

PI Regulator



Parallel realization of the PI algorithm

z^{-1} = unity delay

$k'_i = k_i \cdot T$ (T is the sampling period)

PI Regulator

Given the simplicity of the PI controller equations, the computation of y can be very fast. If we have a mC or DSP with MAC instruction, the algorithm may require only 3 clock cycles:

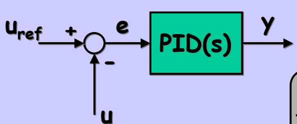
1. accumulator precharge with $y_I(k-1)$;
2. computation of $y_I(k)$, i.e. MAC $e(k), k'_I$;
3. computation of $y(k)$, i.e. MAC $e(k), k_p$;

In the end, the accumulator contains y(k).

Of course, several things may go wrong in the process (overflow, quantization, …)!

PID Regulator

In the analog domain a PID regulator is described by an input-output relation like the following:



$$\frac{Y(s)}{E(s)} = k_p + \frac{k_i}{s} + s \cdot k_d$$

k_i = integral gain

k_p = proportional gain

k_d = derivative gain

PID Regulator

A purely derivative control action cannot be implemented in the analog domain (the corresponding transfer function is not proper), nevertheless it is possible to generate it numerically, for instance like this:

$$y_d(k) = k'_d \cdot [e(k) - e(k-1)], \qquad k'_d = k_d/T$$

The derivative action is very noise sensitive, actually it is a good noise amplifier.

We must use it with great care: a typical provision is to combine it with a series low pass filter.

PID Regulator

The derivative action is normally implemented according to the following algorithm:

$$y_d(k) = \frac{k_d}{T+\tau_L} \cdot [e(k) - e(k-1)] + \frac{\tau_L}{T+\tau_L} \cdot y_d(k-1)$$

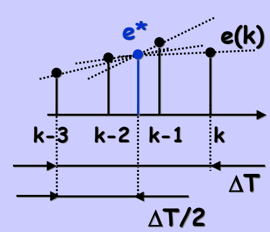
that corresponds to the following continuos time domain transfer function:

$$\frac{Y_d(s)}{E(s)} = \frac{k_d \cdot s}{1 + s \cdot \tau_L}$$

Low pass filter: limits the derivative action at high frequencies.

PID Regulator

As an alternative, we can use more complex derivative algorithms, based on the linear interpolation of several samples.



We create a virtual sample e* that is located at one half of the considered interval (4 samples, here) and whose value is the average of the considered samples.

PID Regulator

The derivative is then expressed as the average value of the incremental ratios computed among the considered samples and the virtual sample e*, that is:

$$\frac{de}{dt} \cong \frac{1}{4}\cdot\left[\frac{e(k)-e^*}{1.5T} + \frac{e(k-1)-e^*}{0.5T} - \frac{e(k-2)-e^*}{0.5T} - \frac{e(k-3)-e^*}{1.5T}\right]$$

where $e^* = \frac{1}{4}\cdot[e(k)+e(k-1)+e(k-2)+e(k-3)]$

We then find:

$$\frac{de}{dt} \cong \frac{1}{6T}\cdot[e(k)+3e(k-1)-3e(k-2)-e(k-3)]$$

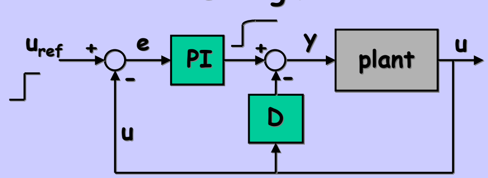
PID Regulator

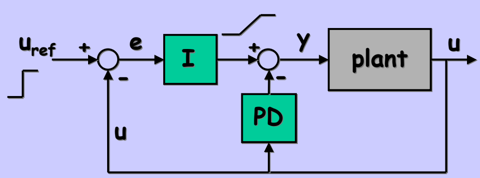
It is possible to extend the average to a bigger number of samples, thus further reducing the sensitivity of the computation to noise. But, in this case, the speed of response becomes lower.

Extending the average to more than a few samples, as in our example, is often not advantageous.

Moreover, it is possible to use different configurations of the PID regulator, where the derivative action is treated differently from the proportional and integral ones.

PID Regulator





PI regulator with anti-wind-up

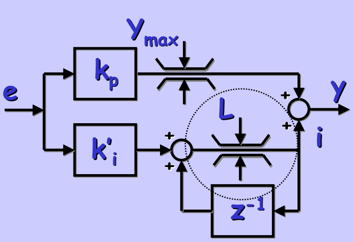
A serious problem with integral regulators is given by the integrator saturation during transients (or in the presence of other saturations in the system control loop).

The presence of a non-zero error at the integrator input for relatively long periods, unavoidably causes undesired desaturation transients, when the regulator comes back to normal operation.

This transient is often unacceptable. It can be removed, if we use a specific provision called anti-wind-up action.

PI regulator with anti-wind-up

The simplest way to operate the anti-wind-up action is the following.



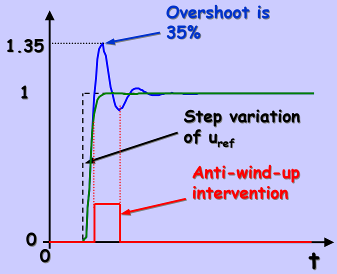
$|L(k)| = Y_{max} - |k_p \cdot e(k)|$

Each control period, we compute limit L. The current output of the integral controller is limited within ±L. This way y is always < Y_{max} in absolute value.

PI regulator with anti-wind-up

The anti-wind-up action complicates the PI algorithm quite a lot, since it needs the evaluation of L, that is of the difference between Y_{max} and the integral controller output at every control cycle. Besides, the limitation of the integral controller requires its comparison with limit L, and, depending on the result, different actions, i.e. the program will include conditional branches. Some mCs allow to reduce this complexity because their assembly include specifically designed instructions (e.g. saturated arithmetic).

PI regulator with anti-wind-up



Unity step responses of a closed loop system with 2 PI regulators. The first does not include anti-wind-up, the second instead does.

The anti-wind-up reduces the overshoots in the reference step variation responses.

Predictive Regulators

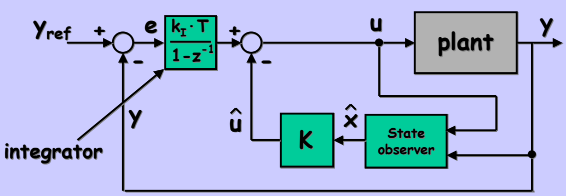
When a reliable model of the controlled plant is available, it is possible to implement predictive or dead-beat controllers.

These can only be implemented digitally, because they are based on a on-line plant model running internally to the controller.

The closed loop system dynamics (in terms of step response), in case of a perfect, ideal model (no model errors), can be made equal to those of a pure delay, of a certain minimum order.

Predictive Regulators

The general structure of a predictive regulator is the following:



The controller needs an estimation of the system state variables, that is used to close the control loop.

Predictive Regulators

Even if it is a very powerful control tool, predictive control is rarely used in industrial applications.

This is due to its relatively high complexity and to the consequent difficulty in the design of the controller parameters.

Moreover, reliable plant models are not always available (in this case, system identification is required).

Lastly, this type of controllers are relatively noise sensitive. Great care must be taken in signal conditioning.