Digital Implementation of Control Laws

Åström, Karl Johan

1984

PRACTICAL ASPECTS ON DIGITAL IMPLEMENTATION OF CONTROL LAWS

K.J. Aström

Department of Automatic Control
Lund Institute of Technology
Box 725, 220 07 Lund 7
Sweden

SUMMARY

Practical problems associated with digital computer implementation control laws are discussed. The key problem is to convert a digital control law in state space or polynomial form into a computer program which gives the desired results. The paper covers: sensor and actuator interfaces, analog prefiltering, actuator saturation, anti-windup, numerics and coding.

1. INTRODUCTION

This paper deals with practical aspects on implementation of digital control laws. The starting point is a description of a control algorithm in terms of a linear dynamical system either in state space form or in transfer function form. A summary of these form is given in Section 2. Analog prefiltering is a necessity when realising digital control laws. This is discussed in Section 3. The consequences of the dynamics of the prefilters and of the computational delay is also covered in this Section. Although many control laws can be designed using linear theory it is necessary to take nonlinearities into account in the implementation. The special case of actuator saturation which is very common is discussed in Section 4. Consequences of roundoff and finite word-length in the calculations are discussed in Section 6. A more detailed treatment of the topics of this paper is given in [1].

2. DIFFERENT REPRESENTATIONS OF THE REGULATOR

Linear design methods give control laws in the form of linear dynamical system. Such systems can be represented in many different ways.

State feedback with an explicit observer

Pole placement or LQG design result in a control law of the form

$$\begin{cases} \hat{x}(k|k) = \hat{x}(k|k-1) + K[y(k) - \hat{y}(k|k-1)] \\ u(k) = L[x_m(k) - \hat{x}(k|k)] + D_c u_c(k) \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \\ x_m(k+1) = f(x_m(k), u_c(k)). \\ \hat{y}(k+1|k) = C\hat{x}(k+1|k) \end{cases} \tag{1}$$

where $\hat{x}$ is an estimate of the process state and x_m is the state of the model which generates the desired response to command signals u_c. Notice that a nonlinear model for the desired state may be used in this representation.

General State Representation

If the function f in (1) is linear the regulator given by (1) is a linear system with the inputs y and u_c and the output u. Such a regulator may always be represented as

$$\begin{cases} u(k) = Cx(k) + Dy(k) + D_c u_c(k) \\ x(k+1) = Fx(k) + Gu(k) + G_c u_c(k). \end{cases} \tag{2}$$

This form is more compact than (1). The state may, however, not necessarily have a simple physical interpretation.

Transfer_Function_models

Several design methods result in a description of the regulator in terms of a linear transfer function model. The general form of such a model can be written as

$$R(q)\,u(k) = T(q)\,u_c(k) - S(q)\,y(k), \qquad (3)$$

where R(q), S(q) and T(q) are polynomials in the forward shift operator q.

There are simple transformations between the different representations of the regulators.

How_to_implement_a_discrete_time_system

The implementation of a discrete time system described by (2.1), (2.2) or (2.3) using a digital computer is straightforward. The computer code for implementation of the regulator given by equation (2.2) is:

```
     Procedure Regulate
     begin
1      Adin y  uc
2      u := C*x + D*y + Dc * uc
3      x := F*x + G*y + Gc * uc
4      Daout u
     end
```

Analog to digital conversion is commanded in the first line. The appropriate values are stored in the arrays y and uc. The control signal u is computed in the second line using matrix vector multiplication and vector addition. The state vector x is updated in the third line, and the digital to analog conversion is performed on line four. To obtain a complete code it is also necessary to have type declarations for the vectors u, uc, x and y and the matrices F, G, Gc, C, D and Dc. It is also necessary to assign values to the matrices and the initial value for the state x. When using computer languages which do not have matrix operations it is necessary to write appropriate procedures for generating matrix operations using operations on scalars.

The details depend on the hardware and software available. To show the principles it is assumed that the system described by (2) should be implemented using a digital computer with A-D and D-A converters and a real time clock. The execution of the program is controlled by the clock. which initiates the execution of the code at each clock interrupt. The sampling period is thus determined by the time between the clock pulses.

It is thus straight forward to implement a digital control law. To obtain a good control system it is however necessary to also consider: numerics, sensors, actuators, operational aspects and programming aspects. These will be discussed in the following sections.

3. PREFILTERING AND COMPUTATIONAL DELAY

To obtain a satisfactory digital system it is necessary to filter the analog signals before they are sampled. It is also necessary to consider the dynamics caused by the prefilter and the computational delay.

Analog_prefiltering

To avoid aliasing it is necessary to use an analog prefilter for elimination of disturbances with frequencies higher than the Nyquist frequency associated with the sampling rate. In signal processing applications the analog prefilter is determined frequency content of the signal, see [2] and [3]. In a control problem there is normally much more information available about the signals in terms of differential equations for the process models and possibly also for the disturbances. An analog Kalman filter would be a very good prefilter, because it can be based on a detailed description of the signal. There are several advantages in implementing the Kalman filter in a computer. In such a case it is useful to sample the analog signals at a comparatively high rate and to avoid aliasing by an ordinary analog prefilter designed from the signal processing point of view.

The bandwidth ω_B of the prefilter is inversely proportional to the sampling period h. A common rule of thumb is to choose the sampling period so that $\omega_B h \approx 0.5 - 1$.

The precise choice depends on the order of the filter and on the character of the measured signal. The dynamics of the prefilter should be taken into account when designing the system.

If the sampling rate is changed the prefilter must also be changed. With reasonable component values it is possible to construct analog prefilters for sampling periods shorter than a few seconds. For slower sampling rates it is often simpler to sample once per second or faster with an appropriate analog prefilter and apply digital filtering to the sampled signal. This approach also makes it possible to change the sampling period of the control calculations by software only.

Since the analog prefilter has dynamics it is necessary to include the filter dynamics in the process model. If the prefilter or the sampling rate is changed it is necessary to recompute the coefficients of the control law.

Crude estimates indicate that with normal sampling rates, like 10-20 times per period, it is indeed necessary to consider the prefilter dynamics.

Computational delay

Since A-D and D-A conversions and computations take time, there will always be a delay when a control law is implemented using a computer. The delay, which is called the computational delay, will depend on how the control algorithm is implemented. There are basically two different ways to do this. The measured variables read at time t_k may be used to compute the control signal to be applied at time t_{k+1}. This is called case A. Another possibility, case B, is to read the measured variables at time t_k and to make the D-A conversion as soon as possible.

The first scheme has the disadvantage that the control actions are delayed unnecessarily and second scheme has the disadvantage that the delay will be variable depending upon the programming. In both cases it is necessary to take the computational delay into account when computing the control law. This is easily done by including a time delay of h or τ respectively in the process model. Another practical detail is that there is a good rule to read the inputs before the outputs are set out. If this is not done there is always the risk of electrical cross coupling.

The computational delay can be made as small as possible by making as few operations as possible between the A-D and D-A conversions.

Consider the previously given program. Since the control signal u is available after executing the second line of code the D-A conversion can be done before the state is updated. The delay may be reduced further by also calculating the product C*x after the D-A conversion. The following algorithm is then obtained.

```
      Procedure Regulate
      begin
1       Adin  y  uc
2       u := u1 + D*y + Dc*uc
3       Daout u
4       x := F*x + G*y + Gc*uc
5       u1 := C*x
      end
```

It is useful to have good estimates of computing times for different control algorithms. A good way to obtain these is to run test programs. For linear control laws it is often possible to estimate times from results of a scalar product computation.

On simple microcomputers, which do not have floating point arithmetic in hardware, there will be a substantial difference in computing time between fixed point and floating point operations. The difference is much less if there is hardware for floating point operations.

To judge the consequences of computational delays it is also useful to know the sensitivity of the closed loop system with respect to a time delay. This may be evaluated from a root locus with respect to a time delay. A simpler way is to evaluate how much the closed loop poles change when a time delay of one sampling period is introduced.

Detection of outliers and measurement malfunctions

Linear filtering theory is very useful to reduce the influence of measurement noise. There may, however, also be other types of errors like instrument malfunctions and conversion errors. These are typically characterized by large deviations which occur with low probabilities. It is, of course, very important to try to eliminate such errors so that large errors do not enter into the control law calculations. There are many good ways to achieve this when using computer control.

The errors may be detected at the source. In systems with high reliability requirements this is done by duplication of the sensors. Two sensors are then combined with a simple logic, which gives an alarm if the difference between the sensor signals is larger than a threshold. A pair of redundant sensors may be regarded as one sensor, which either gives a reliable measurement or a signal that it does not work.

In more extreme cases three sensors may be used. A measurement is then accepted as long as two out of the three sensors agree (two-out-of-three logic). It is of course also possible to use even more elaborate combinations of sensors and filters.

It is also possible to use a Kalman filter for error detection. Consider for example the control algorithm (1) with an explicit observer. The one step prediction error

$$\varepsilon(k) = y(k) - \hat{y}(k|k-1) = y(k) - C\hat{x}(k|k-1)$$

appears explicitly in the algorithm. If estimates of the covariance matrix of the prediction error are available it is easy to test if a particular measurement is reasonable, see [4].

One possibility to obtain the error covariance is to update the covariance equation of the Kalman filter on line.

Kalman filters and redundant sensors pairs may also be combined. If measurement errors are checked in this way it is possible to obtain a very flexible system. The scheme should be augmented with tests to ensure observability. It is thus possible to obtain a system which can provide diagnosis of sensor errors.

Notice that the possibilities of making these types of test depend crucially on the fact that the representation of the control law (1) with an explicit observer is used.

In computer control there are also may other possibilities to detect different types of hardware and software errors. A few extra channels in the A-D converter, which are connected to fixed voltages, may be used for testing and calibration. By connecting a D-A channel to an A-D channel the D-A converter may also be tested and calibrated. The computer may be checked by performing calculations calculations whose results are known and compare the results with the known values.

4. NONLINEAR ACTUATORS

Although linear theory has a wide applicability there are often some nonlinearities which must be taken into account. Actuators often have a saturation characteristics. This nonlinearity may be important when large changes are made. There may be difficulties with the control system during start up and shut down as well as during large changes if the nonlinearities are not considered.

The rational way to deal with the saturation is to develop a design theory which takes the nonlinearity into account. This can be done using optimal control theory. Such a design method is, however, quite complicated. The corresponding control law is also complex. It is therefore practical to use simple heuristic methods.

The reason for the difficulties is that the regulator is a dynamical system. When the control variable saturates it is necessary to make sure that the state of the regulator behaves properly. Different ways of achieving this are discussed below.

State space regulators with an explicit observer

Consider first the case when the control law is described as an observer combined with a state feedback (1). The regulator is thus a dynamical system whose state is represented by the estimated state $\hat{x}$ in (1). In this case it is straightforward to see how the difficulties with the saturation may be avoided.

The estimator (1) will give the correct estimate if the variable u in (1) is chosen as the actual control variable u_p. If the variable u_p is measured the estimate given by (1) and the state of the regulator are thus correct even if the control variable saturates. If the actuator output is not measured it can be estimated provided that the nonlinear characteristics is known. For the case of a simple saturation the control law can thus be written as

$$\begin{cases} \hat{x}(k|k-1) = \hat{x}(k|k-1)+K[y(k)-C\hat{x}(k|k-1)] = [A - KC]\,\hat{x}(k-1|k-1) + B\hat{u}(k-1) \\ \hat{u}_p(k) = \text{sat}\{L[x_m(k) - \hat{x}(k|k)] + u_m\} \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + B\hat{u}(k) \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + B\hat{u}_p(k) \end{cases} \qquad (4)$$

$$\text{sat } u = \begin{cases} u_{low} & u \le u_{low} \\ u & u_{low} < u < u_{high} \\ u_{high} & u \ge u_{high} \end{cases}$$

for a scalar and

$$\text{sat } u = \begin{bmatrix} \text{sat } u_1 \\ \text{sat } u_2 \\ . \\ . \\ . \\ \text{sat } u_n \end{bmatrix}$$

for a vector. The values ulow and uhigh are chosen to correspond to the actuator limitations. Notice that even if the transfer function from y to u for (1) is unstable the state of the system (4) will always be bounded if the matrix A-KC is stable. It is also clear that $\hat{x}$ will be a good estimate of the process state even if the value saturates provided that ulow and uhigh are chosen properly.

The general state space model

The regulator may also be specified as a state space model of the form (2)

$$x(k+1) = F\,x(k) + G\,y(k) \tag{5}$$

$$u(k) = C\,x(k) + D\,y(k) \tag{6}$$

which does not include an explicit observer. The command signals have been neglected for simplicity. If the matrix F has eigenvalues outside the unit disc and the control variable saturates it is clear that windup may occur. Assume for example that the output is at its limit and there is a control error y. The state and the control signal will then continue to grow although the influence on the process is restricted because of the saturation.

To avoid the difficulty it is desirable to make sure that the state of (5) assumes the proper value when the control variable saturates. In conventional process controllers this is accomplished by introducing a special tracking mode which makes sure that the state of the system corresponds to the input output sequence $\{u_p(k), y(k)\}$. The design of a tracking mode may be formulated as an observer problem. In the case of state feedback with an explicit observer the tracking is done automatically by providing the observer with the actuator output u_p or its estimate $\hat{u}_p$. In the regulator given by (5) and (6) there is no explicit observer. To get a regulator which avoids the windup problem the solution for the regulator with an explicit observer will be imitated. The control law is first rewritten as indicated in Fig. 1. The systems in a) and b) have the same input-output relation. The system S_B is also stable. By introducing a saturation in the feedback loop in b) the state of the system S_B is always bounded if y and u are bounded.

This argument may formally be expressed as follows. Multiply (6) by K and add to (5). This gives

$$\begin{aligned} x(k+1) &= F\,x(k) + G\,y(k) + K[u(k) - C\,x(k) - D\,y(k)] \\ &= [F-KC]\,x(k) + [G-KD]y(k) + K\,u(k) \\ &= F_0 x(k) + G_0 y(k) + K\,u(k). \end{aligned}$$

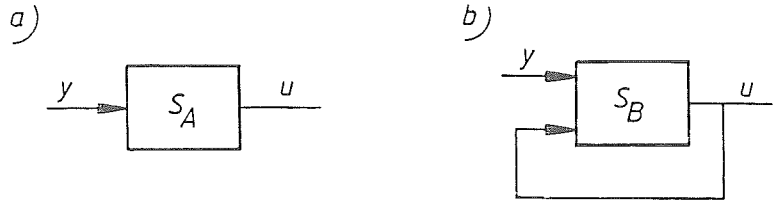

Fig. 1 Different representations of the control law.

If the system (5), (6) is observable the matrix K can always be chosen so that $F_0 = F - KC$ has prescribed eigenvalues inside the unit disc. Notice that this equation is analogous to (4). Applying the same arguments as for the regulator with an explicit observer the control law becomes

$$\begin{aligned} x(k+1) &= F_0 x(k) + G_0 y(k) + K u(k) \\ u(k) &= \text{sat } [Cx(k) + D y(k)]. \end{aligned} \tag{7}$$

The saturation function is chosen to correspond to the actual saturation in the actuator. A comparison with the case of an explicit observer shows that (7) corresponds to an observer with dynamics given by the matrix F_0. The system (7) is of course also equivalent to (2) for small signals.

Transfer Function Form

The corresponding constructions can also be carried out for regulators characterized by input-output models. Consider a regulator described by

$$R(q)\, u(k) = T(q)\, u_c(k) - S(q)\, y(k) \tag{8}$$

where R, S and T are polynomials in the shift operator. The problem is to rewrite the equation so that it looks like a dynamical system with the observer dynamics driven by three inputs, the command signal u_c, the process output y and the control signal u. This is accomplished as follows.

Let $A_o(q)$ be the desired characteristic polynomial of the observer. Adding $A_o(q)u(k)$ to both sides of (8) gives

$$A_o u = T u_c - S y + (A_o - R) u$$

A regulator with anti-windup compensation is then given by

$$\begin{cases} A_o v = T u_c - S y + (A_o - R) u \\ u = \text{sat } v. \end{cases} \tag{9}$$

This regulator is equivalent to (8) when it does not saturate. When the control variable saturates it can be interpreted as an observer with dynamics given by the polynomial A_o.

A particularly simple case is the case of a dead beat observer i.e. $A_0^* = 1$. The model can then be written as

$$u(k) = \text{sat } [T^*(q^{-1})\, u_c(k) - S^*(q^{-1})\, y(k) + (1 - R^*(q^{-1}))\, u(k)] \tag{10}$$

5. NUMERICS

When implementing a computer control system it is necessary to answer questions like: How accurate converters are needed? What precision is required in the computations? Should computations be made in fixed point or floating point arithmetic? To answer these questions it is necessary to understand the effects of the limitations and to estimate their consequences for the closed loop system. This is not a trivial question, because the result will depend on a complex interaction of the feedback, the algorithm and the sampling rate. The real issues fortunately involves crude questions like 10 or 12 bit resolution, 24 or 32 bit wordlength. Such questions may be answered using simplified analysis. A detailed treatment is given in [5].

Error sources

The major error sources are

- Quantization in A-D converters.
- Quantization of parameters.
- Round-off, overflow, and underflow in addition, subtraction, multiplication, division, function evaluation and other operations.
- Quantization in D-A converters.

Common types of A-D converters have accuracies of 8, 10, 12 and 14 bits which corresponds to a resolution of 0.4 %, 0.1 %, 0.025 % and 0.006 %. The percentages are in relation to full scale. The D-A converters have also a limited precision. An accuracy of 10 bits is typical. The error due to the quantization of the parameters will depend critically on the sampling period and on the chosen realization of the control law.

Word-length

Digital control algorithms are typically implemented on micro and minicomputers which have word-lengths of 8, 16 or 32 bits. Special purpose computers where the word-length may be chosen freely are used in applications like the space shuttle or in special products which are made in very large quantities.

There are many differences in number representations. The following representations are common.

- Fixed point single precision 16 bit
- Fixed point double precision 32 bit
- Floating point single precision 8 bit exponent 24 bit mantissa
- Floating point single precision 8 bit exponent 56 bit mantissa

A key problem is that floating point operations are neither associative nor distributive.

Overview of effects of round-off and quantization

An overview of the effects of round-off and quantization will now be given. Tools for analysing the effects will also be discussed.

The consequences of round-off and quantization depend on the feedback system and on the details of the algorithm. The properties may be influenced considerably by changing the representation of the control law or the details of the algorithm. It is thus important to understand the phenomena.

A detailed description of round-off and quantization leads to a complicated nonlinear model which is very difficult to analyse. Investigation of very simple cases shows, however, that quantization and round-off may lead to limit cycle oscillations, see [6] and [7].

Some properties of quantization and round-off in a feedback system may also be captured by linear analysis. Quantization and round-off are then modeled as ideal operations with additive or multiplicative disturbances. The disturbance may be either deterministic or stochastic. This type of analysis is particularly useful for order of magnitude estimation. It allows investigation of complex systems and it is useful when comparing different algorithms, see [8] and [9].

Techniques from sensitivity analysis and numerical analysis are also useful to find the sensitivity of algorithms to changes of parameters. Such methods may be used to compare and screen different algorithms. The methods are, however, limited to comparison of the open-loop performances of the algorithms. It is of course also necessary to compare the effects of quantization and round-off with the other disturbances in the system.

Different realizations

A control law is a dynamical system. Different realizations may be obtained by transforming the state space coordinates. The choice of a suitable realization is very important for the conditioning. In particular the companion forms are very bad from a numerical point of view, see [10]. It is much better to represent a system as a combination of first and second order systems.

If the dynamical system representing the regulator has nr distinct real poles and nc complex pole pairs the control algorithm may be transformed to the model form

$$
\begin{cases}
z_i(k+1) = \lambda_i z_i(k) + \beta_i y(k) \quad i = 1,\dots,nr \\
v_i(k+1) = \begin{bmatrix} \sigma_i & \omega_i \\ -\omega_i & \sigma_i \end{bmatrix} v_i(k) + \begin{bmatrix} \gamma_{i1} \\ \gamma_{i2} \end{bmatrix} y(k) \quad i = 1,\dots,nc \\
u(k) = Dy(k) + \sum_{i=1}^{nr} \gamma_i z_i(k) + \sum_{i=1}^{nc} \delta_i^T v_i(k)
\end{cases}
\tag{11}
$$

where the complex poles are represented using real variables. Notice that z_i are scalars and v_i are vectors with two elements.

To avoid numerical difficulties the control law should thus be transformed into the form (6.4) which is then implemented in the control computer. The transformation may easily be done in a package for computer aided design. Notice that it is easy to use fixed point calculations and scaling for equations in the form (6.4).

If the control law has multiple eigenvalues a Jordan canonical form replaces (6.4). An eigenvalue λ of multiplicity 3 thus corresponds to a block

$$z(k+1) = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} z(k) + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} y(k).$$

Effects of the sampling period

The sampling period also has a considerable influence on the conditioning as is shown by the following examples.

EXAMPLE - Effect of sampling period on coefficient precision

Consider a first order system with time constant T. The discrete time equivalent of such a system is

$$x(t+h) = ax(t) + bu(t),$$

where

$$a = e^{-h/T}.$$

Simple calculations show that

$$\frac{dT}{T} = -\frac{T}{h}\frac{da}{a}.$$

For a given relative precision in the equivalent time constant is thus inverse proportional to the sampling period.

6. CONCLUSIONS

Implementation of control laws using a computer have been discussed in this paper. The key problem is to implement a discrete time system. The principles for doing this have been covered in detail. It was shown that it is straightforward to obtain the code from the control algorithm. The importance of prefiltering to avoid aliasing has been mentioned. Nonlinear digital filtering for removing outliers has also been discussed. It has been mentioned that the computational delay is influenced considerably by the organization of the computer code. Difficulties which arise from saturation in actuators and ways to avoid the difficulties have been discussed. This will also automatically give a solution to mode switching and initialization. Numerical problems and consequences of finite word-length have also been discussed. It was found to be very beneficial to transform the equations describing the control law to a form which is numerically well conditioned. Although the presentation is kept fairly brief the information given should be sufficient to implement control algorithms on mini and micro computers using high level languages.

7. REFERENCES

[1] K.J. Aström and B. Wittenmark: Computer-controlled Systems, Prentice Hall, Englewood Cliffs, N.J, 1984.

[2] F.F. Kuo: Network Analysis and Synthesis, Wiley, New York, 1962.

[3] A.B. Williams: Electronic Filter Design Handbook, McGraw Hill, 1981.

[4] A. Willsky: A survey of design methods for failure detection in dynamic systems. Automatica 12 (1976) 601-611.

[5] P. Moroney: Issues in the Implementation of Digital Feedback compensators. MIT Press, 1983.

[6] S.R. Parker and S.F. Hess: Limit cycle oscillations in digital filters. IEEE Trans. Cirucit Theory CT-18 (1971) 687-697.

[7] L.B. Jackson: Limit cycles in state-space structures for digital filters. IEEE Trans. Circuits & Systems CAS-26 (1979) 67-68.

[8] A.V. Oppenheim and R.W. Schafer: Digital Signal Processing. Prentice Hall, New Jersey, 1975.

[9] L.R. Rabiner and B. Gold: Theory and Application of Digital Signal Processing. Prentice Hall, New Jersey, 1975.

[10] G. Björk, A. Dahlqvist and N. Andersson: Numerical Methods. Prentice Hall, Englewood Cliffs, 1974.