Theory and application of error analysis for improving the performance of practical digital controllers

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/13882

Publisher: © Michael Andrew Oliver

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
Pilkington Library

<table>
<thead>
<tr>
<th>Author/Filing Title</th>
<th>OLIVER M. A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accession/Copy No.</td>
<td>040152565</td>
</tr>
<tr>
<td>Vol. No.</td>
<td></td>
</tr>
<tr>
<td>Class Mark</td>
<td></td>
</tr>
<tr>
<td>- 7 FEB 2000</td>
<td>Loan copy</td>
</tr>
<tr>
<td>29 JUL 2001</td>
<td></td>
</tr>
</tbody>
</table>
Theory and application of error analysis for improving the performance of practical digital controllers

by

Michael Andrew Oliver

A doctoral thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

April 1997

© by Michael Andrew Oliver 1997
Dedicated
to my family

and to the memory of
Dr John K. Prall
SYNOPSIS

The thesis begins with an overview of the four types of error that are generated in digital filters which arise from two sources. The first source is the quantisation of time through the sampling process which gives rise to the first error type - algorithmic error. The second source of error is the quantisation of amplitude due to finite wordlength effects. Coefficient representation error (coefficient error), ADC quantisation error (ADC error) and multiple-word truncation error (truncation error) are all effects of amplitude quantisation.

The shift operator commonly used in digital filters is prone to sensitivity problems especially at high sampling rates. Alternative filter operators are reviewed and advantages and disadvantages of their use are identified in enhancing the performance of digital filters.

The measurement of algorithmic error and coefficient error can be carried out deterministically whilst ADC error and truncation error need to be treated stochastically. The thesis presents a technique to separate the deterministic error and the stochastic error in a composite output signal from a digital filter that is implemented using a high-level language with floating-point arithmetic. This technique represents a method for validating the error analysis techniques.

The error analyses presented in the thesis are used as tools to improve the performance of emulated digital filters. Each different error form varies in an individual manner with respect to changes in the sampling period. By careful selection of a sampling rate the filter performance can be improved by ensuring that the magnitudes of all the error forms are consistent with each other.

Similar techniques can be used to improve the performance of digital compensators in closed-loop control systems. Careful consideration of the duration of the sampling interval needs to be observed to ensure that the stability of the closed-loop system is adequate. This method of determining the controller parameters is used to exploit the digital control hardware to its fullest potential, and is automated in software; details are described in the thesis.
ACKNOWLEDGEMENTS

I am indebted to my supervisor, Dr Bill Forsythe for his guidance, encouragement and enthusiasm throughout the duration of the research.

I would also like to express sincere gratitude towards The Engineering and Physical Sciences Research Council (EPSRC) and Loughborough University (of Technology) for financially making this project possible.

I would like to thank all members of the Systems and Control Research Group, both past and present, for their useful discussions and advice. These people include: Dr Abdullah El-Abbar, Dr Mustafa 'Reg' Abuzied, Edwin Foo, Andy Fox, Dr Malcolm Fraser, John Gow, Dr Robert Habib, Dr Peter Holme, Li Hong, Dr Kenneth Nai, Dr John Paddison, Dr Dipesh Patel, Dr John Pearson, Dr Ian Pratt, Reza Sheikhan and Peter Gang Shen. Their positive sense of humour maintained high morale within the group!

Thanks must also be given to Mr Sardev Singh for the loan of a digital camera and for 'developing' the digital photographs of the practical set-up.

Finally, I would like to thank my family and Sarah for their encouragement and support throughout the last three and a half years.
"Improving the performance of emulated digital filters",
M.A. Oliver & W. Forsythe,
Irish Digital Signal Processing And Control Colloquium, IDSPCC '95.
Queen's University Of Belfast,
pp 339 - 346,
June 1995.

"Causes and effects of algorithmic error in digital filters",
M.A. Oliver & W. Forsythe,
Control Engineering Practice, Vol 3, No. 10,
pp 1421 - 1432,
October 1995.

"The delta operator : a comparative study",
M.A Oliver & W. Forsythe,
Irish Digital Signal And Control Conference, IDSPCC '96.
Trinity College, Dublin,
pp 125 - 132,
June 1996.
## LIST OF PRINCIPAL SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_i)</td>
<td>digital filter numerator coefficients ((z)-filter)</td>
</tr>
<tr>
<td>(b_i)</td>
<td>digital filter denominator coefficients ((z)-filter)</td>
</tr>
<tr>
<td>(c_i)</td>
<td>digital filter numerator coefficients ((\delta_z)-filter)</td>
</tr>
<tr>
<td>(e_{\text{adc}})</td>
<td>ADC quantisation error</td>
</tr>
<tr>
<td>(e_q)</td>
<td>rms quantisation error</td>
</tr>
<tr>
<td>(f_i)</td>
<td>digital filter numerator coefficients ((\delta_m)-filter)</td>
</tr>
<tr>
<td>(g_i)</td>
<td>digital filter denominator coefficients ((\delta_m)-filter)</td>
</tr>
<tr>
<td>(j)</td>
<td>(\sqrt{-1})</td>
</tr>
<tr>
<td>(q)</td>
<td>quantisation noise</td>
</tr>
<tr>
<td>(p_i, r_i)</td>
<td>poles and residues from a partial expansion of a transfer function (Chapter Four)</td>
</tr>
<tr>
<td>(r_i)</td>
<td>digital filter denominator coefficients ((\delta_z)-filter)</td>
</tr>
<tr>
<td>(s)</td>
<td>Laplace operator</td>
</tr>
<tr>
<td>(t)</td>
<td>time (seconds)</td>
</tr>
<tr>
<td>(u(.))</td>
<td>input signal</td>
</tr>
<tr>
<td>(u_q(.))</td>
<td>quantised input signal</td>
</tr>
<tr>
<td>(y(.))</td>
<td>output signal</td>
</tr>
<tr>
<td>(w)</td>
<td>(w)-operator (for (w)-plane controller design)</td>
</tr>
<tr>
<td>(z)</td>
<td>(z)-operator</td>
</tr>
<tr>
<td>(A(z))</td>
<td>numerator of an ideal (z)-filter</td>
</tr>
<tr>
<td>(A^*(z))</td>
<td>numerator of a (z)-filter with quantised coefficients</td>
</tr>
<tr>
<td>(B(z))</td>
<td>denominator of an ideal (z)-filter</td>
</tr>
<tr>
<td>(B^*(z))</td>
<td>denominator of a (z)-filter with quantised coefficients</td>
</tr>
<tr>
<td>(E_a)</td>
<td>algorithmic error (global)</td>
</tr>
<tr>
<td>(E_c)</td>
<td>coefficient representation error</td>
</tr>
<tr>
<td>(E_d)</td>
<td>quantised-coefficient algorithmic error (also referred to as deterministic error)</td>
</tr>
<tr>
<td>(E_l)</td>
<td>algorithmic error (local)</td>
</tr>
<tr>
<td>(E_m)</td>
<td>multiple-word truncation error</td>
</tr>
</tbody>
</table>
\( E_p \) proportional algorithmic error
\( E_s \) step algorithmic error (Chapter 2)
\( E_s \) stochastic error
\( F(s) \) continuous filter or control system
\( G(z) \) digital filter with exact coefficients
\( H(z) \) digital filter with quantised coefficients
\( \text{IN} \) inverse normalised DFT
\( M \) magnitude
\( \text{ND} \) normalised DFT
\( [Q], [R] \) matrices used to calculate \( \Phi_0 \)
\( [S] \) matrix of sensitivity factors
\( T \) sampling interval (seconds)
\( Z(\cdot) \) "take the z-transform of \( \{\cdot\}\)"
\( \delta, \delta_e \) delta-operator (Goodall's definition)
\( \delta_m \) delta-operator (Middleton & Goodwin's definition)
\( \sigma_{q^2} \) variance of output quantisation noise
\( \phi \) phase (radians)
\( \omega \) angular frequency (radians per second)
\( \Phi_0 \) autocorrelation function of output

All other symbols are introduced and defined in the text.
# TABLE OF CONTENTS

Synopsis

i

Acknowledgements  ii

Publications  iii

List of principal symbols  iv

Table of contents  vi

Chapter One: Introduction

1

1.1  Structure of the thesis  4

1.2  Background material  5

1.2.1  Emulation techniques  5

1.2.2  Digital controller design methods  8

1.2.3  The delta operator  9

1.2.4  Wordlength requirements  10

1.3  Objectives  11

Chapter Two: Algorithmic Error

2

2.1  Introduction  12

2.2  A time-domain measure of algorithmic error  14

2.2.1  Example  15

2.3  A frequency-domain measure of algorithmic error  15

2.3.1  Example  16

2.3.2  Variation of algorithmic error with signal and sampling frequency  17

2.4  Determining algorithmic error using inverse emulation  18

2.4.1  Example  23

2.5  Algorithmic global, local and step errors  26

2.5.1  Global error, $E_g$  26

---

vi
2.5.2 Local error, \( E_l \)
2.5.3 Step error, \( E_s \)
2.5.4 Relationships

2.6 A comparison of emulation techniques

2.7 Proportional error, \( E_p \)
2.7.1 Using proportional error as a means of determining sampling frequency

2.8 Summary

Chapter Three: Coefficient representation error

3.1 Introduction
3.1.1 Variation of coefficient representation with signal and sampling frequency

3.2 Fixed-point and floating-point binary representations
3.2.1 Fixed-point format
3.2.2 Floating-point representation
3.2.3 Comparing fixed-point and floating-point representations

3.3 The mechanism of coefficient representation error
3.3.1 Floating-point representation
3.3.2 Fixed-point representation

3.4 Coefficient sensitivity

3.5 Quantised-coefficient algorithmic error

3.6 Summary

Chapter Four: Quantisation noise analysis

4.1 Introduction
4.1.1 Introduction to rounding error
4.1.2 Introduction to truncation error

4.2 Methods of estimating the effects of roundoff noise
4.2.1 Worst-case error bound 66
4.2.2 Steady-state worst-case error bound 67
4.2.3 R.M.S. estimate 68
4.2.4 A comparison of the three estimates 70
4.3 Numerical methods for evaluating the contour integral 73
  4.3.1 Cauchy's residue theorem 73
  4.3.2 Jury's numerical method 75
  4.3.3 Åström's numerical method 77
4.4 A corrected method to numerically evaluate the contour integral 78
  4.4.1 Simple poles 79
  4.4.2 Second-order systems with double pole 80
  4.4.3 Third-order systems with double pole 81
  4.4.4 Third-order systems with triple pole 82
  4.4.5 Transfer functions having complex poles 82
4.5 Examples 83
  4.5.1 The effectiveness of the new technique 83
  4.5.2 A comparison of methods for computing the contour integral 85
4.6 Summary 87

Chapter Five: A comparison of alternative filter operators 93
  5.1 Introduction 93
  5.2 Algorithmic error 94
  5.3 Coefficient representation error 95
  5.4 ADC quantisation error 97
  5.5 Multiple-word truncation error 98
  5.6 Coefficients 101
  5.7 Speed of computation 102
  5.8 The modified canonic filter architecture 102
  5.9 Summary 104
Chapter Six: Hardware design

6.1 Introduction 113
6.2 Digital controller design 114
  6.2.1 PC interface 114
  6.2.2 Timer 114
  6.2.3 Digital controller hardware 115
  6.2.4 Digital controller software 116
6.3 Plant Models 118
  6.3.1 Model of a closed-loop system 118
  6.3.2 Second-order lightly-damped system 119
  6.3.3 Second-order Type I system 120
6.4 Testing 120
6.5 Summary 121

Chapter Seven: Digital filter error analysis using D.F.T. techniques 128

7.1 Introduction 128
7.2 Spectral analysis 130
7.3 A normalised DFT algorithm 132
  7.3.1 Example: Computing the NDFT 133
7.4 Definitions 134
7.5 Analysis 135
  7.5.1 Total error definition 135
  7.5.2 Deterministic error 138
  7.5.3 Evaluating stochastic error 139
7.6 Examples 140
7.7 Summary 142

Chapter Eight: Improving the performance of digital filters 148

8.1 Introduction 148
8.2 Quantised-coefficient algorithmic error 148
  8.2.1 Example 149
8.3 Using a figure of merit to optimise filter performance
  8.3.1 Type I figure of merit
  8.3.2 The choice of T
  8.3.3 Example 1
  8.3.4 Example 2
  8.3.5 An alternative figure of merit (Type II)
  8.3.6 Example 3
  8.3.7 Example 4
  8.3.8 Example 5
  8.4 Internal variable considerations
    8.4.1 Determining the basic wordlength
    8.4.2 Overflow requirements
    8.4.3 Underflow requirements
  8.5 Software design
  8.6 Summary

Chapter Nine: Improving the performance of digital controllers
184
  9.1 Introduction
  9.2 Digital design techniques
    9.2.1 Emulation
    9.2.2 W-plane design techniques
      9.2.2.1 w-plane design
      9.2.2.2 w'-plane design
    9.2.3 Comparisons
  9.3 Improving the performance of single-loop control systems
    9.3.1 Choosing a value for T
    9.3.2 Wordlength requirements
  9.4 Software design
  9.5 Practical examples
9.5.1  Examples 1 & 2  196
9.6  Other digital control topologies  197
    9.6.1  Digital control with the delta-operator  197
    9.6.2  Example 3  197
    9.6.3  Example 4  198
    9.6.4  Two-loop control strategies using first-derivative feedback  199

    9.6.5  Example 5  200
    9.6.6  Two-loop control strategies controlling two physical systems  201
    9.6.7  Example 6  202
9.7  Summary  203

Chapter Ten: Conclusion  222
    10.1  Original contributions  223
    10.2  Recommendations for further work  224

References  225
Bibliography  229

Appendix A: Infinite Series  230
Appendix B: Quantisation noise analysis for systems with multiple poles  239
Appendix C: Circuit diagrams  247
Appendix D: Analogue compensator design software  252
Appendix E: MATLAB source code for the digital control design software  265
Before the widespread availability of digital computers, filters and compensators were realised using analogue components. Analogue filter circuits can be constructed entirely from passive components such as resistors and capacitors, but active components such as transistors, and later operational amplifiers, could be included to add gain. The design of analogue circuits is straightforward and the cost of the components is relatively inexpensive. Despite the ease of construction for analogue filter and compensator circuits, careful adjustments must be made to the circuit in order for it to fulfil the required function. In analogue circuits, the component values tend to 'drift' with time, especially with temperature variations, causing the function of the filter to deviate from that intended.

With the advent of the digital computer, the notion of realising filters and controllers digitally became a reality. The required hardware, which comprises, amongst other devices, a processor, read-only and random access memory, analogue-to-digital and digital-to-analogue converters, is relatively expensive when compared with the components required to implement analogue filters. However, due to advances in modern technology, the cost of digital processing devices is consistently dropping, making digital control a very attractive alternative.
As the parameters of a digital controller or filter are programmed into the system, any required modifications can be made by altering the filter software. In addition, unlike analogue filters, the coefficients of a digital filter do not drift. A major advantage that a digital control system has over analogue control is that single-chip designs are possible, reducing PCB intricacy and cost for complex control strategies.

Unfortunately, the output of a digital filter will always differ from that of its analogue counterpart from which it is derived, assuming both are subjected to the same input signal. The differences between the two are caused essentially by two types of quantisation: the quantisation of time and the quantisation of amplitude. Consequently, four types of error are inherent within digital filters:

(i) algorithmic error,
(ii) coefficient representation error,
(iii) ADC quantisation error, and
(iv) multiple-word truncation error.

The first of these types of error arises due to the quantisation of time; the rest are amplitude quantisation effects.

Algorithmic error is the error introduced when a digital filter is derived from a continuous design. It is strongly dependent on the emulation technique by which the digital filter is derived and the sampling rate.

In order to store the coefficients of a practical digital filter, the exact coefficient in an ideal digital filter requires shortening to fit into a finite wordlength. The discrepancy between the outputs of the practical and ideal digital filters constitutes coefficient representation error.

In practice the input signal to the digital controller will be subjected to quantisation as a result of the finite wordlength of the input sampler. The resulting error, which can be represented as noise, propagates through the digital filter as ADC quantisation error.
The final source of error arises when the product of a multiplication is reduced to a lesser length. This resulting quantisation noise leads to multiple-word truncation error.

One technique of minimising the difference in behaviour between an analogue and a digital filter would be to use a fast sampling rate for the digital filter. In order to carry this out successfully long coefficient wordlengths would be required. If acceptable provision for the coefficient wordlengths is not made then the advantage gained by fast sampling will be eliminated by inadequate coefficient representation. Conversely the performance of a digital filter having long coefficients could be compromised if the sampling rate is too low.

Therefore, in order to obtain the best possible performance from a digital filter for given hardware, trade-offs should be made between the choices of sampling rate and coefficient wordlength. This procedure would ensure that the errors caused by coefficient inaccuracy and the sampling process itself would be of similar orders of magnitude thus minimising the overall error.

A state-of-the-art digital signal processor with facilities for fast sampling and long coefficient wordlengths would invariably provide the foundation for a digital filter or controller whose performance would not differ greatly from the analogue equivalent. Despite the potential performance of such a system, the hardware could be costly. A question then arises: 'Could adequate performance be achieved at lower cost?' For a processing device offering lower computational capability, provided care is taken to select an appropriate range of sampling speeds for a given coefficient wordlength, the answer to the question, in most cases, is 'Yes'. Further considerations could then be taken to select a specific sampling period to further optimise the performance of the filter. The choice of a less powerful processing device may result in a saving of only a few pounds for a single controller or filter, but this will increase several hundred or thousand-fold if the filter hardware is mass produced.
1.1. Structure of the thesis

The Thesis is divided into ten chapters. Each chapter has been written to be as self-contained as possible that dependence on previous chapters is minimal.

Chapter Two presents a detailed analysis of causes and effects of algorithmic error in digital filters. The theory in this chapter is not solely constrained to digital signal processing and control applications, but it could be valuable for other fields, for example economics and forecasting, where a digital filter may be employed.

Chapter Three discusses coefficient representation error and investigates the trends in variations of the error with respect to changes in the sampling rate.

Quantisation noise analysis, which is discussed in Chapter Four, can be applied to analyze both ADC quantisation error and multiple-word truncation error. The content of Chapter four reviews established techniques for estimating the effects of quantisation noise on the outputs of digital filters, along with the correction and development of an existing technique.

Two definitions for the delta operator exist and this has led to differences in opinion between control engineers concerning which operator should be used. Chapter Five makes a comparative study between the two definitions.

Chapter Six discusses the design of hardware which is utilised at a later stage of the thesis to verify in practice most of the theoretical aspects of the research. Simple digital control hardware is designed in which the sampling period may be manually adjusted. Several simple analogue filters are also designed in the Chapter to mimic the behaviour of physical systems.

The error analysis covered in chapters Two to Four investigates, in isolation, how the various error forms affect the output of a digital filter. In practice all the error forms
coexist. Chapter Seven presents a technique, which uses a modification of the Discrete Fourier Transform, to separate and isolate the deterministic and stochastic components of the error from the overall output signal. The results of this technique can be used to verify the error analysis detailed in chapters Two to Four.

Chapter Eight presents a novel technique for determining a value for the sampling interval which can drastically improve the performance of a digital filter with respect to its analogue counterpart. This is achieved by making trade-offs between the various error types, and practical demonstrations illustrate the benefits of using this technique.

The work presented in Chapter Eight is extended in Chapter Nine to the optimisation of closed-loop digital control systems with respect to the analogue equivalent. The effectiveness of this technique is also verified using practical demonstrations.

Finally, general conclusions may be found in Chapter Ten which also discusses possible recommendations for further work.

The remainder of the introduction briefly reviews some relevant background material.

1.2. Background material

This section reviews very briefly the background material used and referred to in the body of the text.

1.2.1. Emulation techniques

The majority of the Thesis is concerned with attempting to match the performance of a digital filter, as closely as possible, to that of an equivalent analogue filter. The process of emulation is the usual process of translating a digital filter from the continuous domain to the discrete. Many methods of emulation exist and can essentially be classified into three broad categories:
Introduction

(i) differentiation methods,
(ii) integration methods, and
(iii) mapping methods.

The simplest methods of performing differentiation are to find the gradient between two successive samples of a signal. The forwards and backwards rectangular rules [Fr90] are differentiation methods which respectively replace each Laplace operator 's' in the continuous transfer function with

\[ s = \frac{z - 1}{T} \]  

or

\[ s = \frac{1 - z^{-1}}{T} \]

where T is the sampling interval of the digital filter.

Of these two first-order differentiation methods, the backwards rule gives better emulation. Further improvements in the emulation can be achieved by using a higher-order differentiation algorithm such as those discussed in [Fo79]. Even though higher order differentiation algorithms improve the emulation, the order of the resulting digital filter will be higher than the original continuous filter.

An alternative to substituting 's' by a differentiation algorithm is to use an integration algorithm. The simplest method of performing integration, despite being relatively ineffective, is to use rectangular integration. Conversely, trapezoidal integration, which also avoids computational complexity, is an extremely effective emulation technique. This emulation method, which was published by Tustin in 1947, is very popular and is applied in most digital control texts, e.g. [Fr90, Fo91]. Tustin's transformation is often referred to as the bilinear transform and is given by
The order of the denominator in certain analogue filters is often higher than that of its numerator. The application of the bilinear transform usually results in a digital filter whose numerator and denominator are of equal order. This could lead to large discrepancies between the analogue and digital time-domain responses during the transient phase. To overcome this problem, Janisowski [Ja93] presented a modification to the bilinear transform which ensures that the denominator of the digital transfer function in the $z$-plane is of higher order than the numerator. This modification is carried out by cascading the transfer function

$$J(z) = \frac{2z^{-1}}{1 + z^{-1}}$$

with the discrete transfer function produced from Tustin's transformation. Although this method increases the order of the denominator it does not upset the d.c. gain.

The bilinear transform gives a digital transfer function whose d.c. gain is equal to that of the equivalent continuous filter. In order to match the continuous and discrete frequency responses at another frequency point other than d.c. a technique which involves the bilinear transform with 'prewarping' can be applied [Fo91].

The final category of emulation techniques to be discussed is the classification by mapping methods. By observing that the relationship between the Laplace operator, $s$, and the shift operator, $z$, is given by

$$z = \exp(sT)$$

the poles and zeros of the s-domain transfer function may be mapped into the $z$-domain. In addition, a gain factor also needs to be added to match the d.c. gains of the continuous and discrete filters.
A problem arises when the zeros of the continuous filter lie at the point \(-\infty\) in the s-plane. Such zeros are usually mapped to the points \(z = -1\) or \(z = 0\) in the discrete plane; however both methods are unsatisfactory. This issue was tackled by Forsythe [Fo85a, Fo85b, Fo91] and led to a new method in which the poles of the continuous filter are mapped to the z-domain but the zeros are determined by a Taylor series expansion for each \(z^1\) term.

1.2.2. Digital controller design methods

It is possible to design a digital controller directly using digital design techniques [Kn94]. However the indirect design of a digital controller via classical control design is often preferred. This subsection reviews some techniques for designing digital controllers indirectly using classical design methods that most control engineers are familiar with.

The simplest digital control design method is emulation [Fo91]. The first stage of the design process is to devise an analogue compensator to satisfactorily control a physical system in a closed-loop feedback system. For a chosen sampling period, the transfer function of the digital controller is derived directly using an emulation technique, usually the bilinear transform. For fast sampling rates, emulation can be a successful control strategy. However, for slow sampling rates, the effect of the zero-order hold (ZOH), situated at the output of the digital controller, causes system destabilisation, resulting in unsatisfactory responses.

A more successful design technique is w-plane design [Le85, Fo91]. For this technique a ZOH model of the plant is determined to take into account the phase lag introduced by the zero-order hold. This model is translated, using a bilinear transformation, to a pseudo-continuous plane, known as the w-plane. The application of classical control design methodologies for the w-plane plant model results in the design of a continuous w-plane controller. The required digital controller is found, using a bilinear transform, from the w-plane controller.
1.2.3. The delta operator

In order to reduce the problems of coefficient sensitivity associated with filters realised using the shift operator, two different alternatives, both known as delta, have been suggested.

One version, developed by Goodall [Go89, Fo91, Go93] is defined as

$$\delta_g = z - 1.$$  \hfill (1.6)

A general second-order \( z \)-filter

$$G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$  \hfill (1.7)

takes the form

$$G(\delta_g) = \frac{c_0 + c_1 \delta_g^{-1} + c_2 \delta_g^{-2}}{1 + r_1 \delta_g^{-1} + r_2 \delta_g^{-2}}$$  \hfill (1.8)

in the \( \delta_g \) plane. It can be shown that the numerator coefficients of Goodall's delta filter are related to the numerator coefficients of the \( z \)-filter by

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$  \hfill (1.9)

The above relationship can also be applied to the denominator coefficients. The inverse relationship can be used to convert the coefficients of a delta filter to the \( z \)-plane.

Another version of delta, developed by Middleton [Mi86, Mi90] is defined as

$$\delta_m = \frac{z - 1}{T}.$$  \hfill (1.10)

A general second-order Middleton delta filter takes the general form
\[
G(\delta_m) = \frac{f_0 + f_1 \delta_m^{-1} + f_2 \delta_m^{-2}}{1 + g_1 \delta_m^{-1} + g_2 \delta_m^{-2}}
\]

(1.11)

and the z-domain coefficients are related to the \( \delta_m \) coefficients by

\[
\begin{bmatrix}
1 & 0 & 0 \\
2/T & 1/T & 0 \\
1/T^2 & 1/T^2 & 1/T^2
\end{bmatrix}
\begin{bmatrix}
a_0 \\ a_1 \\ a_2
\end{bmatrix}
\]

(1.12)

1.2.4. Wordlength requirements

All the values in a practical digital filter, whether coefficient values or internal variables, need to be accommodated in fixed wordlengths. If due care is not exercised when determining wordlength requirements, the performance of the resulting practical digital filter could be seriously impaired. This subsection reviews some of the existing methods for determining coefficient and variable wordlengths in digital filters.

The coefficients of a practical digital filter will require some quantisation in order for them to fit into a predefined number of bits. The coefficients, and therefore the performance of the digital filter, will therefore differ from that intended. A deterministic technique of sensitivity analysis [Go89, Fo91, Go93, Go94] has been developed to facilitate the determination of wordlength requirements for the coefficients of digital filters. Using this technique a matrix of sensitivity coefficients is determined which relates fractional changes of a digital filter coefficient to fractional changes of equivalent continuous filter coefficients. With knowledge of how changes in digital coefficients relate to the continuous coefficients, the correct number of bits can be allowed for the coefficient wordlengths such that specified accuracy requirements in the analogue coefficients can be met.

The internal variables of a digital filter consist of three components:

(i) the basic wordlength
Introduction

(ii) some overflow allowance, and
(iii) some underflow allowance.

These three components must be selected carefully in order for the digital filter to respond in the appropriate manner to various inputs.

The basic wordlength is defined by the number of bits associated with the analogue-to-digital converter (ADC). Clearly, the filter performance will be potentially enhanced for long ADC wordlengths or smaller quantisation step sizes.

The size of the variables in a digital filter can increase to many times the size of the input. The provision of several overflow bits [Fo91] is required such that the maximum possible input does not cause the internal variables to saturate.

The internal variables in a digital filter are determined by the product of a coefficient by another variable; the wordlength of the resulting variable will be equal to the sums of wordlengths of both the coefficient and other variable. To prevent the length of the internal variable growing indefinitely, it will have to be reduced to a lesser length. This has the effect of generating quantisation noise. In order to minimise the effects of this noise, and to ensure the filter responds properly to small inputs, a provision of several underflow bits must be made.

1.3. Objectives

The main aim of the research is threefold:

(i) To identify and quantify the effects of all sources of error inherent in digital filters and digital controllers.
(ii) To use the error analysis to select a suitable value for the sampling interval for the digital filter or controller which reduces the overall error with respect to an analogue equivalent reference.
(iii) To implement hardware for the practical verification of the theoretical work.
CHAPTER TWO
ALGORITHMIC ERROR

2.1 Introduction

Various methods are described in text books and research papers for generating a digital filter which will emulate the performance of a given analogue filter, normally without comment on the effectiveness of the methods. This chapter investigates among other things the problem of comparing methods of emulation. Two distinct types of error are inherent in digital filters: quantisation error and algorithmic error. This chapter concerns itself with algorithmic error and highlights techniques which can be used to evaluate this type of error given a digital transfer function and the analogue counterpart from which it was derived. It shows how the observed algorithmic error is dependent upon the emulation technique used and the sampling interval of the digital filter. Algorithmic error analysis is shown to be a useful tool for determining a lower bound on the sampling frequency of the filter. Considerations of local error and step error show how the global algorithmic error is generated.

There has been a considerable amount of research carried out upon the various types of quantisation error which occur in digital filters but there has been comparatively little research performed on the algorithmic aspect; notable exceptions being [Fo86] and [Ol95b].
The type of error which is widely referred to as 'quantisation error' is an effect of amplitude quantisation. Such errors arise as a result of analogue-to-digital conversion, the representation of filter coefficients to a finite precision, and multiple word truncation in mathematical operations. An overview of such error analysis may be found in [Li71] and in a number of digital control texts, e.g. [Fo91], [Fr90] and [Ka81].

Algorithmic error is caused by the quantisation of time, as a consequence of sampling at fixed instants. As intuition would suggest, when the sampling interval of a digital filter is reduced, the effects caused by the quantisation of time also diminish. It is widely recognised, in addition, that such errors are heavily dependent on the emulation technique used to derive the digital filter or discrete algorithm - hence the name algorithmic error.

Algorithmic error is defined as the difference at each sampling instant between the outputs of a continuous filter and an equivalent infinite precision digital filter. This situation is illustrated in Figure 2.1.

When evaluating algorithmic error for a particular digital filter, all quantisation effects are avoided, that is, infinite precision is approximated in the arithmetic by using long wordlengths. The architecture chosen to realize the filter is therefore of no consequence.

This chapter investigates three techniques by which algorithmic error may be determined. It also demonstrates how the magnitude of the error varies according to the sampling interval, the input signal frequency, for the case of sinusoids, and the emulation technique by which the digital filter is derived.

For all the analyses performed in this Chapter and the rest of the text, it may be assumed that the Nyquist Limit is not exceeded.
2.2. A time-domain measure of algorithmic error

Referring back to Figure 2.1, algorithmic error is defined as the difference between the output of a continuous filter and the output of the corresponding digital filter at the instants of sampling. The following analysis yields the time history of the algorithmic error expressed as a z-transform. Representing the error in terms of a z-transform is a convenient notation in so far as the error value is only valid at the sampling instants.

Also, with reference to Figure 2.1, let the input to the system be $u(t)$ and the output of the analogue filter be $y(t)$ where their transforms are

$$y(s) = u(s)F(s).$$

(2.1)

Expressing this waveform as the z-transform of a sequence of its samples gives

$$y(z) = Z\{y(s)\}.$$  

(2.2)

For the digital section of the model shown in Figure 2.1, the input after sampling is $u(z)$. Let the output of the digital section be $y'(z)$ where

$$y'(z) = u(z)G(z).$$

(2.3)

The algorithmic error is the difference between the output of the continuous filter and the output of the digital filter at each sampling instant. From equations (2.1) to (2.3), it follows that the algorithmic error, $E_a(z)$ is

$$E_a(z) = y(z) - y'(z) = Z\{u(s)F(s)\} - u(z)G(z)$$

(2.4)

The expression that equation (2.4) produces is the z-transform of the algorithmic error expressed in a so-called 'closed-form', as a ratio of two polynomials expressed in terms of $z$. Performing long division on this expression will produce the sequence of error values.
2.2.1 Example

Consider the second-order continuous filter.

\[ F(s) = \frac{10(s + 1)}{s^2 + s + 10}. \]  

(2.5)

Using the bilinear transform and a sampling interval of \( T = 0.5 \) seconds, the corresponding digital transfer function is

\[ G(z) = \frac{1.6667 + 0.6667z^{-1} - z^{-2}}{1 - 0.4z^{-1} + 0.7333z^{-2}}. \]

(2.6)

Assuming that the input \( u(t) \) is a unit step, then from equations (2.1) and (2.2),

\[ y(z) = Z\left\{ \frac{10s + 10}{s^2 + s + 10} \right\} \]

(2.7)

and

\[ y'(z) = \frac{1}{1 - z^{-1}} \cdot \frac{1.6667 + 0.6667z^{-1} - z^{-2}}{1 - 0.4z^{-1} + 0.7333z^{-2}}. \]  

(2.8)

Inserting these results into equation (2.4) yields the z-transform of the algorithmic error. Figure 2.2 illustrates how the algorithmic error varies in response to a unit step. This waveform is confirmed by subtracting the time-domain waveforms.

2.3. A frequency-domain measure of algorithmic error

This section describes an accurate, and computationally efficient means, of calculating the peak algorithmic error in the case of a sinusoidal signal.

The term 'residual' is used in this thesis, referring to the stage of time-domain analysis when all the transients of the output signal have died away. This is often referred to as the 'a.c. steady-state' in the case of sinusoidal excitation.
Referring to Figure 2.1, let the input signal be sinusoidal with unity magnitude, and angular frequency, \( \omega \).

\[ u(t) = \sin \omega t. \]  

(2.9)

By making the substitution \( s = \text{j} \omega \) in \( F(s) \), the magnitude and phase of the residual output from the continuous section can be obtained as

\[ M_f \angle \phi_f = F(\text{j} \omega) \]  

(2.10)

where \( M_f \) is the magnitude of the output signal from \( F(s) \), and \( \phi_f \) is the phase.

By making the substitution \( Z = \text{exp}(\text{j} \omega T) \) into the digital filter, \( G(z) \), the magnitude and phase of the residual stage digital output can be computed as

\[ M_z \angle \phi_z = G(\text{j} \omega T). \]  

(2.11)

Thus, the peak algorithmic error at this frequency is the magnitude of the difference between the two output signals. That is

\[ E_a(\omega, T) = |M_f \angle \phi_f - M_z \angle \phi_z| \]  

(2.12)

The variation of algorithmic error with time will be sinusoidal.

2.3.1 Example

Consider again the lightly-damped continuous transfer function

\[ F(s) = \frac{10 (s + 1)}{s^2 + s + 10}. \]  

(2.13)

Using the bilinear transformation with \( T = 0.5 \) seconds, as before, leads to
\[ G(z) = \frac{1.6667 + 0.6667z^{-1} - z^{-2}}{1 - 0.4z^{-1} + 0.7333z^{-2}}. \] (2.14)

Let the input to the system be a unity magnitude sinusoid of angular frequency \( \omega = 0.1 \) radians per second. The magnitude and phase of the output from the continuous transfer function can be obtained from equation (2.10) as

\[ M \phi_f = 1.005943159 \angle 0.089658976. \] (2.15)

Similarly the magnitude and phase of the output of the digital filter is found using equation (2.11) to be

\[ M \phi_g = 1.005945633 \angle 0.089677519. \] (2.16)

From equation (2.12) the peak algorithmic error is, for this example,

\[ E_a(0.1, 0.5) = 1.88 \times 10^{-5}. \] (2.17)

Figure 2.3 shows the difference between the outputs of \( F(s) \) and \( G(z) \) during the residual stage of simulation following an input of the unit sinewave described above. This is the actual algorithmic error. The graph also shows the line corresponding to the peak algorithmic error, \( E_a \) as calculated.

### 2.3.2 Variation of algorithmic error with signal and sampling frequency

A simple first-order phase advance network, whose continuous transfer function is

\[ F(s) = 4.76 \left[ \frac{2.11s + 1}{0.48s + 1} \right] \] (2.18)

may be transformed into \( G(z) \) using the bilinear transform. The algorithmic error varies with both signal and sampling frequencies as illustrated in Figure 2.4.

From Figure 2.4, it may be observed that the magnitude of algorithmic error decreases
as the signal frequency is reduced or the sampling frequency is increased. Both of these effects result in more samples being taken per cycle of the input signal, to yield greater accuracy, as would be expected.

However, for certain filters possessing very sharp roll-off characteristics, algorithmic error will decrease as the signal frequency is increased. This result is also to be expected.

2.4. Determining algorithmic error using inverse emulation

A third technique for determining algorithmic error can again be applied to residual analysis. This method relies on work described in [F083], [F085a], [F085b] and [F086], in which a time delay, $z^{-1}$ is expressed as a power series based upon the Taylor expansion; relevant material is summarised below.

Consider the general digital transfer function

$$G(z) = \frac{y'(z)}{u(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$ (2.19)

which has been derived from the general second-order continuous transfer function $F(s)$

$$F(s) = \frac{y(s)}{u(s)} = \frac{n_0 + n_1 s + n_2 s^2}{m_0 + m_1 s + m_2 s^2}$$ (2.20)

$y(t)$ denotes the output from the continuous filter $F(s)$, while $y'(t)$ represents the output from the digital filter $G(z)$.

Cross-multiplying equation (2.19) and expressing as a function of time gives

$$a_0 u(t) + a_1 u(t-T) + a_2 u(t-2T) = y'(t) + b_1 y'(t-T) + b_2 y'(t-2T)$$ (2.21)

The delayed input terms of equation (2.21) may be expanded using the Taylor series, so that
Algorithmic Error

\[ u(t-T) = u(t) - T \ddot{u}(t) + \frac{T^2}{2!} \dot{u}(t) - \frac{T^3}{3!} u(t) + \ldots \]  
(2.22)

and

\[ u(t-2T) = u(t) - 2T \ddot{u}(t) + \frac{4T^2}{2!} \dot{u}(t) - \frac{8T^3}{3!} u(t) + \ldots \]  
(2.23)

For the general situation where the input is delayed by \( n \) sampling periods

\[ u(t-nT) = u(t) - nT \ddot{u}(t) + \frac{n^2T^2}{2!} \dot{u}(t) - \frac{n^3T^3}{3!} u(t) + \ldots \]  
(2.24)

The same approach can be applied to the delayed output terms of equation (2.21).

The Taylor expansion technique may be illustrated by translating the general case of equation (2.24) into the \( s \)-domain. That is

\[ u(t-nT) \rightarrow u(s) \left\{ 1 - nTs + \frac{(nTs)^2}{2!} - \frac{(nTs)^3}{3!} + \ldots \right\} \]  
(2.25)

As the number of bracketed terms approaches infinity, the bracketed expression tends towards \( e^{-nTs} \), or \( z^{-n} \) - a delay of \( n \) sampling intervals.

The left hand side of equation (2.21) can be rewritten in terms of \( z \) and expanded using the Taylor series to give the general form:

\[ u(z) \left\{ a_0 + a_1z^{-1} + a_2z^{-2} \right\} \rightarrow q_0u(t) + q_1\dot{u}(t) + q_2\ddot{u}(t) + q_3\dddot{u}(t) + \ldots \]  
(2.26)

where the \( q_i \) coefficients can be determined using
Algorithmic Error

\[
\begin{bmatrix}
1 & 1 & 1 \\
0 & -T & -2T \\
0 & \frac{T^2}{2!} & \frac{4T^2}{2!} \\
0 & \frac{-T^3}{3!} & \frac{-8T^3}{3!} \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\vdots \\
\end{bmatrix} =
\begin{bmatrix}
q_0 \\
q_1 \\
q_2 \\
q_3 \\
\vdots \\
\end{bmatrix}
\]  

(2.27)

or

\[V_dA = Q.\]  

(2.28)

\[V_d\] is a matrix derived from the terms in the Taylor series expansions. The depth of the matrix is governed by the number of terms required in the overall error expression. Clearly the more terms used, the more accurate the final result. The width of the \(V_d\) matrix equals the number of \(a\)-coefficients. The \(A\) matrix is a column vector which consists of the numerator coefficients of equation (2.19).

In a similar manner, the right hand side of equation (2.21) may be rewritten in terms of \(z\) and expanded such that in general

\[y'(z)(1 + b_1z^{-1} + b_2z^{-2}) = p_0\ddot{y}(t) + p_1\dot{y}(t) + p_2\dot{y}(t) + p_3\ddot{y}(t) + \ldots\]  

(2.29)

The \(p_i\) coefficients are determined using

\[
\begin{bmatrix}
1 & 1 & 1 \\
0 & -T & -2T \\
0 & \frac{T^2}{2!} & \frac{4T^2}{2!} \\
0 & \frac{-T^3}{3!} & \frac{-8T^3}{3!} \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
1 \\
b_1 \\
b_2 \\
\vdots \\
\end{bmatrix} =
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3 \\
\vdots \\
\end{bmatrix}
\]  

(2.30)

or

\[V_dA = Q.\]  

(2.28)
(2.31)

\[ V_B \mathbf{B} = \mathbf{P} \]

\( \mathbf{B} \) is a column vector consisting of the denominator coefficients of equation (2.19). \( \mathbf{V}_b \) is based around the constructs of \( \mathbf{V}_a \), whose width equals the number of b-coefficients. In this case, the dimensions of the \( \mathbf{V}_a \) and \( \mathbf{V}_b \) matrices are identical, but this is not always the case. The result of the matrix multiplication is stored in the \( \mathbf{P} \) vector.

By defining the following row matrices and dropping the 't' for notational convenience

\[
\mathbf{U} = \begin{bmatrix} u & u & u & \ldots \end{bmatrix} \quad (2.32)
\]

and

\[
\mathbf{Y}' = \begin{bmatrix} y' & y' & y' & \ldots \end{bmatrix} \quad (2.33)
\]

expressions (2.26) and (2.29) can now be written as

\[
u(z)\{a_0 + a_1z^{-1} + a_2z^{-2}\} = \mathbf{UQ} \quad (2.34)\]

and

\[
y'(z)\{1 + b_1z^{-1} + b_2z^{-2}\} = \mathbf{Y}'\mathbf{P}. \quad (2.35)\]

By rewriting equation (2.21) as

\[
y'(z)\{a_0 + a_1z^{-1} + a_2z^{-2}\} = u(z)\{1 + b_1z^{-1} + b_2z^{-2}\} \quad (2.36)\]

it is apparent that

\[
\mathbf{UQ} = \mathbf{Y}'\mathbf{P}. \quad (2.37)\]

It follows that the q- and p-coefficients are respectively the numerator and denominator coefficients of a continuous transfer function which models the discrete transfer function.
By defining the following triangular matrix

\[
\begin{bmatrix}
  p_0 \\
  p_1 & p_0 \\
  p_2 & p_1 & p_0 \\
  p_3 & p_2 & p_1 & p_0 \\
  \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\]

the output signal can be expressed in terms of the input signal and its derivatives as

\[
y'(t) = UP_m^{-1}Q.
\]

The reasoning behind this approach is as follows: Suppose there exists a transfer function

\[
H(s) = \frac{y''(s)}{u(s)} = \frac{q_0 + q_1s + q_2s^2}{p_0 + p_1s + p_2s^2}.
\]

This closed-form expression could be expanded using long division into an infinite series. Thus

\[
H(s) = \frac{y''(s)}{u(s)} = h_0 + h_1s + h_2s^2 + h_3s^3 + h_4s^4 + ...
\]

and therefore

\[
y''(s) = u(s)(h_0 + h_1s + h_2s^2 + h_3s^3 + h_4s^4 + ...).
\]

The matrix equation of expression (2.39) can perform this long division process, and by multiplying the result by the input and its derivatives, the output value of \(y''(t)\) may be determined.
Algorithmic Error

The output signal from the continuous filter $F(s)$ of equation (2.20) may be determined in a similar manner using

$$\gamma(t) = UM_m^{-1}N$$  \hfill (2.43)

where

$$M_m = \begin{bmatrix} m_0 \\ m_1 & m_0 \\ m_2 & m_1 & m_0 \\ 0 & m_2 & m_1 & m_0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$  \hfill (2.44)

and

$$N = [n_0 \ n_1 \ n_2 \ 0 \ \cdots]^T.$$  \hfill (2.45)

Recalling that algorithmic error is the difference between the outputs of the digital and the original continuous filters at each sampling instant, algorithmic error may be approximated using

$$E_g(t) = \gamma(t) - \gamma'(t) = U [M_m^{-1}N - P_m^{-1}Q]$$  \hfill (2.46)

This technique of determining algorithmic error is only suitable for residual analysis. During the transient phase, the continuous model of $G(z)$, defined by the p- and q-coefficients only reflects the action of $G(z)$, when taken to an infinite number of terms.

2.4.1 Example

Consider the second-order digital filter

$$G(z) = \frac{0.2667 + 0.2667z^{-1}}{1 - 0.9333z^{-1} + 0.2000z^{-2}}$$  \hfill (2.47)

which has been derived from the continuous transfer function
Algorithmic Error

\[ F(s) = \frac{s + 4}{s^2 + 3s + 2} \]  \hspace{1cm} (2.48)

using the bilinear transfer function and a sampling interval of \( T = 0.5 \) seconds.

Using the coefficients from the discrete transfer function, the \( A \) and \( B \) matrices can be written as follows:

\[
A = \begin{bmatrix}
    a_0 \\
    a_1 \\
    a_2
\end{bmatrix} = \begin{bmatrix}
    0.2667 \\
    0.2667 \\
    0
\end{bmatrix} \hspace{1cm} (2.49)
\]

and

\[
B = \begin{bmatrix}
    b_1 \\
    b_2
\end{bmatrix} = \begin{bmatrix}
    1 \\
    -0.9333 \\
    0.2000
\end{bmatrix} \hspace{1cm} (2.50)
\]

Extending equation (2.27) to allow for 5 terms, the \( q \)-coefficients are

\[
Q = [q_0 \ q_1 \ q_2 \ q_3 \ q_4]^T
\]

\[
= [0.5333 \ -0.1333 \ 0.0333 \ -0.0056 \ 0.0007]^T. \hspace{1cm} (2.51)
\]

The \( p \)-coefficients can be found in a similar manner using equation (2.30) so that

\[
P = [p_0 \ p_1 \ p_2 \ p_3 \ p_4]^T
\]

\[
= [0.2667 \ 0.2667 \ -0.0167 \ -0.0139 \ 0.0007]^T. \hspace{1cm} (2.52)
\]

By inspection of the continuous transfer function,

\[
N = [n_0 \ n_1 \ n_2 \ 0 \ 0]^T
\]

\[
= [4 \ 1 \ 0 \ 0 \ 0]^T \hspace{1cm} (2.53)
\]
Algorithmic Error

and

\[
M_m = \begin{bmatrix}
m_0 & 0 & 0 & 0 & 0 \\
m_1 & m_0 & 0 & 0 & 0 \\
m_2 & m_1 & m_0 & 0 & 0 \\
o & m_2 & m_1 & m_0 & 0 \\
o & 0 & m_2 & m_1 & m_0
\end{bmatrix} = \begin{bmatrix}
2 & 0 & 0 & 0 & 0 \\
3 & 2 & 0 & 0 & 0 \\
1 & 3 & 2 & 0 & 0 \\
0 & 1 & 3 & 2 & 0 \\
0 & 0 & 1 & 3 & 2
\end{bmatrix}
\]  

(2.54)

Now, let the input to the system be

\[ u(t) = t^4 \]  

(2.55)

chosen because it has a finite number of derivatives.

Using equation (2.32), the input matrix, \( U \), is

\[ U = \begin{bmatrix}
t^4 & 4t^3 & 12t^2 & 24t & 24
\end{bmatrix}. \]  

(2.56)

By applying these results to equation (2.46), after the \( P_m \) matrix is created, the algorithmic error is given by

\[ E_a(t) = U \begin{bmatrix} 0 & 0 & -0.0521 & 0.1146 \end{bmatrix}^T \]  

(2.57)

or

\[ E_a(t) = -1.2504t + 2.7504. \]  

(2.58)

This gives a convenient means of expressing the algorithmic error for the residual state.

Figure 2.5 shows the plot of this error curve and the actual simulated algorithmic error. It can be seen that this method of calculating algorithmic error corresponds with the simulated error after the transients have decayed.

If the input is taken to be a sinusoid and its derivatives therefore known, then algorithmic error can be calculated rather than measured. A sufficient number of terms...
needs to be used, however, to ensure accuracy.

Considering the system used in the example of Section 2.3.1, the coefficients from the continuous and discrete filters are used to form the matrices \( N, M_m, A \) and \( B \). Calculating the algorithmic error using this inverse emulation technique in response to a unity sinewave of frequency 0.1 radians per second results in the response of Figure 2.6. The infinite series in which the algorithmic error could be expressed is truncated to just the first five terms. A comparison of Figure 2.6 with Figure 2.3, where the peak algorithmic error is \( 1.88 \times 10^{-5} \), verifies the validity of the reverse emulation technique for residual analysis.

### 2.5. Algorithmic global, local and step errors

Consider the first-order differential equation

\[
y(t) + m_1 y(t) = n_1 u(t) + n_1 \ddot{u}(t)
\]  

(2.58)

and its representation in difference form as

\[
y'(t) = a_2 u(t) + a_1 u(t-T) - b_1 y'(t-T)
\]  

(2.59)

where \( y(t) \) represents the output of the differential equation and \( y'(t) \) represents the output of the difference equation in response to an input \( u(t) \).

#### 2.5.1 Global Error, \( E_g \)

If the current value of \( y'(t) \) is computed from equation (2.59) using the previous value of \( y'(t) \) as the current value of \( y'(t-T) \) - the normal process - then the difference

\[
E_g(t) = y(t) - y'(t)
\]  

(2.60)

is the current value of global algorithmic error.
2.5.2 Local Error, $E_l$

Now define
\[ y''(t) = a_2u(t) + a_1u(t-T) - b_2y(t-T). \]  

Equation (2.61) is different from equation (2.59) only in that $y(t-T)$ is used in place of $y'(t-T)$. All the terms on the right of equation (2.61) are error-free, yet $y''(t)$ will be in error by
\[ E_l(t) = y(t) - y''(t). \]

$E_l(t)$ is the local error - the error generated in one iteration of the algorithm of equation (2.61).

2.5.3 Step Error, $E_s$

This is simply the difference between successive values of global error, and therefore is defined
\[ E_s(t) = E_g(t) - E_g(t-T) \]  

or in the z-domain
\[ E_s(z) = E_g(z) (1 - z^{-1}) \]  

and therefore
\[ E_s(z) = \frac{E_g(z)}{1 - z^{-1}}. \]

2.5.4 Relationships

Even though local algorithmic error is not directly apparent when making the comparison between two successive values of global algorithmic error, it does play an important role in the mechanism of global error accumulation. Step error and local
error are therefore not the same, but are related as follows:

From equation (2.60)

\[ y(t-T) = y'(t-T) + E_d(t-T). \]  \hspace{1cm} (2.66)

From equation (2.62)

\[ y''(t) = y(t) - E_f(t). \]  \hspace{1cm} (2.67)

Substituting equations (2.66) and (2.67) into equation (2.61) gives

\[ y(t) - E_f(t) = a_o u(t) + a_1 u(t-T) - b_1 [y'(t-T) + E_d(t-T)]. \]  \hspace{1cm} (2.68)

Rearranging equation (2.68) gives

\[ y(t) - [a_o u(t) + a_1 u(t-T) - b_1 y'(t-T)] = E_f(t) - b_1 E_d(t-T). \]  \hspace{1cm} (2.69)

From equation (2.59)

\[ y(t) - y'(t) = E_f(t) - b_1 E_d(t-T). \]  \hspace{1cm} (2.70)

So

\[ E_d(t) = E_f(t) - b_1 E_d(t-T). \]  \hspace{1cm} (2.71)

Writing equation (2.71) as a z-domain expression gives

\[ E_f(z) = E_d(z)(1 + b_1 z^{-1}). \]  \hspace{1cm} (2.72)

The relationship between global algorithmic error and local algorithmic error is given through a transfer function relationship:
Algorithmic Error

\[ E_a(z) = E_r(z) \frac{1}{1 + b_1z^{-1}} \]  \hspace{1cm} (2.73)

From equations (2.65) and (2.72)

\[ E_s(z) = E_r(z) \frac{1 - z^{-1}}{1 + b_1z^{-1}} \]  \hspace{1cm} (2.74)

For the sake of clarity the above argument was developed for a first-order difference equation, but it can clearly be extended to any order. To generalise for \( n \)-th order digital filters, the expressions for global error and step error in terms of local error are respectively

\[ E_a(z) = E_r(z) \frac{1}{1 + b_1z^{-1} + \ldots + b_nz^{-n}} \]  \hspace{1cm} (2.75)

and

\[ E_s(z) = E_r(z) \frac{1 - z^{-1}}{1 + b_1z^{-1} + \ldots + b_nz^{-n}} \]  \hspace{1cm} (2.76)

A technique is given in [Fo86] for evaluating \( E_l \) in terms of the output \( y(t) \), or the input \( u(t) \) and their derivatives: using the terminology introduced in Section 2.4, local error, \( E_l \) can be calculated as

\[ E_l(t) = YV_yB - UV_dA. \]  \hspace{1cm} (2.77)

2.6. A comparison of emulation techniques

Traditionally, the performance of digital filters derived from a continuous filter through various emulation techniques is assessed by making a comparison of each digital frequency response with the analogue response. Results could, in such comparisons, be somewhat misleading. For example the digital magnitude versus frequency plot could correspond well with the equivalent continuous plot, but the phase plot could lack fidelity, or vice-versa. By using algorithmic error as defined in equation (2.12) to
assess the performance of emulation techniques, such ambiguities can be eliminated because the algorithmic error calculations take both magnitude and phase into account.

Consider, for example, the following second-order continuous filter

\[ F(s) = \frac{s + 4}{s^2 + s + 10}. \] (2.78)

From this, using four emulation techniques, four different digital filters can be derived. Let the sampling interval be \( T = 0.5 \) second in each case.

Using a conventional pole and zero mapping technique where the infinite zero of \( F(s) \) is mapped to the point \( z = 0 \) in the \( z \)-plane, denoted PZ0, results in

\[ G_d(z) = \frac{0.7363 - 0.0996z^{-1}}{1 - 0.0149z^{-1} + 0.6065z^{-2}}. \] (2.79)

Using a similar pole and zero mapping technique, but with the infinite zero of \( F(s) \) mapped to the point \( z = -1 \) in the \( z \)-plane, denoted PZ1, gives

\[ G_d(z) = \frac{0.3682 + 0.3183z^{-1} - 0.0498z^{-2}}{1 - 0.0149z^{-1} + 0.6065z^{-2}}. \] (2.80)

The bilinear transform (BT) gives

\[ G_c(z) = \frac{0.2667 + 0.2667z^{-1}}{1 - 0.4z^{-1} + 0.7333z^{-2}}. \] (2.81)

Finally, a technique is considered where poles are mapped but the zeros of the digital filter are derived via the Taylor expansion of the numerator of the continuous transfer function, as described in [Fo83]. This, referred to as modified pole mapping, MPM, gives

\[ G_d(z) = \frac{0.2499 + 0.4854z^{-1} - 0.0985z^{-2}}{1 - 0.0149z^{-1} + 0.6065z^{-2}}. \] (2.82)

Further emulation methods may be found in [Fo91] and [Fr90].
Using the frequency-domain measure of algorithmic error described in Section 2.3, the algorithmic error for all four emulations is illustrated in Figure 2.7 over a range of signal frequencies.

Graph (a) corresponds to the PZM0 technique. This form of emulation produces the greatest algorithmic error, and is therefore the poorest of the four emulation techniques considered. The PZM1 technique illustrated in graph (b) produces less algorithmic error than the previous technique, but the magnitude of the error is still large. The BT and the MPM techniques, represented by plots (c) and (d) respectively, generate the least algorithmic error. The bilinear transform is somewhat better for the lower signal frequencies, but the superiority of the other method is evident for the higher signal frequencies.

To determine visually whether the bilinear transform or the modified pole mapping technique produces the least algorithmic error over ranges of sampling and signal frequency, the ratio of the two algorithmic error quantities needs to be determined.

\[
\text{Ratio} = \frac{E_a(BT)}{E_a(MPM)} \tag{2.83}
\]

Figure 2.8 illustrates the ratio of algorithmic error produced by the bilinear transform to that produced by modified pole mapping for a wide range of signal frequencies and sampling frequencies. Any value less than one suggests that the bilinear transform is the superior of the two techniques for that particular value of sampling frequency and signal frequency. Conversely, a value greater than one for the ratio implies the superiority of the modified pole mapping method.

By plotting these results on a contour-style graph, a boundary may be determined which segregates the regions of sampling and signal frequency in which a particular emulation technique yields the least algorithmic error. Thus, the results of Figure 2.8 are translated onto the contour plot of Figure 2.9. The region below the boundary indicates the range of signal frequencies and sampling frequencies in which the
modified pole mapping produces the least algorithmic error when compared with the bilinear transform. The region above the boundary indicates the range of signal and sampling frequencies where the bilinear transform dominates.

Clearly, it makes sense to use either the bilinear transform, or the modified pole mapping method and it may be wise to compare both techniques over the required range of signal and sampling frequencies.

2.7. Proportional error, $E_p$

When performing detailed error analysis, the raw values of algorithmic error have very little significance. A more useful measure would be to define a sinusoidal error to signal ratio, the proportional error, $E_p$. This may be defined as the magnitude of the algorithmic error over the magnitude of the output of the continuous filter for a given signal frequency. That is

$$E_p = \frac{E_a(\omega,T)}{M_f(\omega)}$$

where $M_f$ is the magnitude of the output of the continuous filter at an input signal frequency, $\omega$, as defined by equation (2.10), and $E_a(\omega,T)$ is the peak algorithmic error value defined by equation (2.12).

Figure 2.10 illustrates the variation of the proportional error for the second-order filter of Section 2.3.1 over a range of varying signal and sampling frequencies. The filter is realised using the bilinear transform.

2.7.1 Using proportional error as a means of determining sampling frequency

Proportional error provides the means for assessing the accuracy of emulation. For digital control based applications, a 1% accuracy in the emulation may be acceptable in so far as the parameters, which define the physical system to be controlled are
Algorithmic Error

seldom known to any greater accuracy. More stringent requirements may be applied for accurate signal processing applications.

Proportional error provides a useful technique of determining the sampling interval, T, of the digital filter. For example, suppose the filter is cascaded with a system having a bandwidth of \( \omega_b \) and an accuracy of 1% is required. The curves of proportional error can be examined at a signal frequency of \( \omega_b \) to determine the minimum sampling frequency that will maintain algorithmic error below 1%. For this example, if the bandwidth of the system to be controlled is 10 radians per second, then by inspection of Figure 2.10, a sampling frequency of approximately 30 Hz results in a proportional error of 1%. So a lower bound for the sampling rate is 30 samples per second; faster sampling would obviously improve the accuracy. This analysis, of course, does not take into account any of the amplitude quantisation effects, which will add to the overall error of the digital filter.

2.8. Summary

This chapter has illustrated several techniques by which algorithmic error may be determined. This type of error decreases as the sampling rate is raised whatever the emulation technique used to derive the digital filter. By making a comparison of algorithmic error over a range of sampling intervals and signal frequencies for various emulation techniques, the superior emulation technique may readily be determined. Of the four emulation methods considered, the bilinear transform and the modified pole mapping methods were clearly better than the well-known pole and zero mapping techniques. Although this has only been demonstrated for one example here, it was found to be the case for all the examples that have been looked at, and so it is believed it to be generally true.

The chapter defines global, local and step error and the relationships between them. Although the local error cannot be directly observed, it is an important concept in understanding global algorithmic error accumulation.
Proportional error is a useful measure which can be used at the filter design stage to give a lower bound on the range of acceptable sampling rates. In practical applications, the effects of amplitude quantisation need to be taken into account for a full error analysis, and this will be illustrated in the forthcoming chapters.

Figure 2.1. Model of algorithmic error generation

Figure 2.2. Algorithmic error variation in response to a unit step
Algorithmic Error

Figure 2.3. Variation of algorithmic error during the residual state.

Figure 2.4. Variation of algorithmic error with sampling frequency and signal frequency.
Figure 2.5. Variation of the theoretical and actual algorithmic error with time.

Figure 2.6. Variation of algorithmic error with time calculated for the system of section 2.3.1 using reverse emulation.
Figure 2.7. Variation of peak algorithmic error with signal frequency for various emulation techniques. (T = 0.5 seconds.)

Figure 2.8. Comparing the ratio of algorithmic error produced by the bilinear transform and the modified pole mapping technique.
Figure 2.9. Comparing emulation techniques. The boundary segregates the regions where a certain emulation produces the least error.

Figure 2.10. Variation of proportional error with sampling rate and signal frequency.
CHAPTER THREE
COEFFICIENT REPRESENTATION ERROR

3.1 Introduction

Coefficient representation error exists in digital filters as a result of specifying the filter coefficients to fixed accuracy. The consequence of quantising the coefficients to 'fit into' the hardware results in a digital filter whose properties differ slightly from those intended. Thus coefficient representation error is defined as the difference at each instant of sampling between the output of an ideal digital filter and the output of the equivalent practical digital filter, whose coefficients have been quantised. (The coefficients of the ideal digital filter are specified to unlimited accuracy.) The model that demonstrates this error mechanism is illustrated in Figure 3.1.

Suppose there exists an ideal discrete transfer function $G(z)$ and an equivalent $H(z)$ whose coefficients have been fixed to some specified wordlength. Let the sampling interval of both filters be $T$ seconds. After both transfer functions have been subjected to the same input $u(t)$, the difference between the outputs of $G(z)$ and $H(z)$ will result in the coefficient representation error. Therefore the $z$-transform of the coefficient representation error is
Coefficient Representation Error

\[ E_c(z) = u(z) \{ G(z) - H(z) \} . \]  

(3.1)

For sinusoidal excitations of angular frequency \( \omega \), the frequency responses of the ideal and realised transfer functions are \( G(j\omega T) \) and \( H(j\omega T) \) respectively. It follows that the peak value of coefficient representation error in response to a unit sinewave is

\[ |E_c(\omega, T)| = |G(j\omega T) - H(j\omega T)| . \]  

(3.2)

As an example consider the second-order continuous filter

\[ F(s) = \frac{10 (s + 1)}{s^2 + s + 10} . \]  

(3.3)

Using the bilinear transform and sampling at 100 times per second, the corresponding ideal digital transfer function is

\[ G(z) = \frac{0.0499875653 + 0.000497339z^{-1} - 0.0494901766z^{-2}}{1 - 1.989057448z^{-1} + 0.9900522258z^{-2}} . \]  

(3.4)

Suppose the coefficients of \( G(z) \) are quantised to be accommodated in fixed-point format which allows for 16 fractional bits then the resulting digital transfer function is

\[ H(z) = \frac{3276 + 33 z^{-1} - 3243 z^{-2}}{65536} \frac{33 z^{-1}}{65536} + \frac{3243 z^{-2}}{65536} . \]  

(3.5)

For a unit input sinusoid of angular frequency \( \omega = 1 \) radian per second the frequency responses of the ideal and practical digital filters are respectively:

\[ G(j\omega T) = 1.21952 + j0.97562 \]  

(3.6)

and

\[ H(j\omega T) = 1.23748 + j0.97677 . \]  

(3.7)

Using equation (3.2) it follows that the peak value of coefficient representation error...
for the given digital filter and input signal is

\[ E_c (1, 0.01) = 0.0180 \]  \hspace{1cm} (3.8)

Figure 3.2 which illustrates the time-domain simulation of the above computation confirms this result.

3.1.1. Variation of coefficient representation error with signal and sampling frequency

The second-order lightly-damped transfer function of equation (3.3) may be transformed into the z-domain using the bilinear transform to form \( G(z) \). The coefficients of \( G(z) \) are then represented to 16 fractional bits; the resulting digital filter being \( H(z) \). Frequency-domain calculation of coefficient representation error using equation (3.2) is carried out for signal frequencies varying from 0.1 to 10 radians per second. Figure 3.3 illustrates how the coefficient representation error varies with respect to the signal frequency for digital filters realised with sampling rates ranging from 10 to 100 samples per second.

It is interesting to compare the plots of coefficient representation error with signal frequency against the magnitude plot of the continuous filter \( F(s) \), which is displayed in Figure 3.4. There is a certain correlation between the plots of coefficient representation error and filter gain, with the peak value of coefficient representation error coinciding with the peak value of gain. For low values of signal frequency, the coefficient representation error decreases and asymptotes to a value which is the difference between the d.c. gains of the two filters. For the case where \( T = 0.0316 \) seconds the difference between the d.c. gains of \( G(z) \) and \( H(z) \) is of the order of \( 10^{-14} \) hence the sharp roll-off of error for low frequencies. For high signal frequencies approaching half the sampling frequency, the expression \( \exp(j \omega T) \) asymptotes towards the value \( \exp(j \pi) = -1 \). The coefficient representation error for the highest possible signal frequency within the Nyquist limit is given by
As the signal frequency is raised then the coefficient representation error decreases and asymptotes towards the value given by equation (3.9).

Using sampling rates varying from 10 to 1000 samples per second, a digital filter can be created from the transfer function of equation (3.3) using the bilinear transform and the coefficients represented to 16 fractional bits. Figure 3.5 presents a three dimensional plot illustrating how coefficient representation error varies with respect to sampling rate and signal frequency. The coefficient representation error varies with respect to signal frequency as mentioned earlier. As the sampling rate is raised the coefficient representation error tends to increase, albeit in a random manner. (This mechanism will be discussed later.)

It can be seen from Figure 3.5 that certain values of sampling frequency lead to 'crevices' on the plot where the coefficient representation error is significantly lower than for its neighbouring sampling points. The performance of a digital filter can therefore be improved by choosing a sampling frequency which lies in a deep 'crevice' rather than on a 'ridge'. Further improvements can be achieved by iteratively searching the region around a chosen crevice to find the deepest part of it. However this only has practical implications if the resulting sampling time, $T$, can be implemented by the digital filter hardware.

### 3.2. Fixed-Point And Floating-Point Binary Representations

Two general forms of arithmetic notation are widely used to store numbers in binary format. These are fixed-point and floating-point representations. This section presents a description of the two formats and compares the merits of using both forms.
3.2.1. Fixed-Point Format

A fixed-point binary representation of a number is comprised of a fixed number (m) of integer bits, a 'binary-point' and a fixed number (n) of fractional bits. This general representation is illustrated in Figure 3.6.

From Figure 3.6 the most significant bit corresponds to $2^m$, the next bit represents $2^{m-1}$ and so on. The first bit following the binary-point represents $2^{-1}$, or 0.5, and the least significant bit represents $2^{-n}$. The largest number that can be represented by such a scheme is $(2^{m+1} - 2^n)$, and the smallest number that can be realised is $2^n$.

The conversion of an ideal, unquantised value $x$ into fixed-point representation may be carried out using [Fo91]

$$x_q = \text{integer} \left( \frac{2^n \times x + 0.5}{2^n} \right)$$  \hspace{1cm} (3.10)

where $n$ is the number of fractional bits. This quantisation scheme employs rounding and the deviation of $x_q$ from its original value $x$ lies within the range

$$-2^{-(n-1)} \leq (x - x_q) < 2^{-(n-1)}.$$  \hspace{1cm} (3.11)

Truncation is another system which may be used to represent fixed-point numbers. In this scheme all bits less than the least significant bit are discarded. The formula for representing a number, $x$, to $n$ fractional bits using truncation is given by [Fo91]

$$x_q = \text{integer} \left( \frac{2^n \times x}{2^n} \right)$$  \hspace{1cm} (3.12)

The deviation of $x_q$ from its original value $x$ lies in the range

$$0 \leq (x - x_q) < 2^{-n}.$$  \hspace{1cm} (3.13)

It can be seen from equations (3.11) and (3.13) that rounding produces a smaller maximum deviation than truncation.
3.2.2. Floating-point representation

A floating-point representation consists of two components; a mantissa and an exponent [Ph80, Os80]. This is illustrated in Figure 3.7. It is usual practice [Ph80] for the mantissa (M) to be normalised such that

\[ 0.5 \times M < 1. \]  \hspace{1cm} (3.14)

For binary floating-point arithmetic, the exponent (E) will take the form \(2^\gamma\) where \(\gamma\) is a signed integer. A positive value of \(\gamma\) shifts the mantissa \(2^\gamma\) places to the left whilst a negative value will shift the mantissa \(2^\gamma\) places to the right. Therefore floating-point arithmetic has a much higher dynamic range than fixed-point arithmetic.

3.2.3. Comparing fixed-point and floating-point representations

As an example, consider a fixed-point arithmetic scheme which represents numbers to 4 fractional bits, and a floating-point format which uses a 4-bit mantissa. The following analysis investigates which number format is appropriate for storing various ranges of numbers. Table 3.1 illustrates how various numerical values are represented using both numerical formats.

The first section of Table 3.1 illustrates the representation of values that are less than or equal to 0.5. For this example, using fixed-point representation with four fractional bits, numbers smaller than 0.0625 cannot be represented because 0.0625 corresponds to the least significant bit. Using floating-point arithmetic with a 4-bit mantissa, numbers less than 0.0625 can be readily represented; all this requires are extra right-shifts of the mantissa.

The second section of Table 3.1 shows how numbers greater than 0.5 and 1 are represented. From the representations there is no difference between the representations of these values using fixed-point and floating-point formats.
Table 3.1. A comparison of fixed-point and floating-point number representations.

The last section in Table 3.1 illustrates the representations of numbers which are greater than 1 using both fixed-point and floating-point formats. From the table, it can be seen that fixed-point arithmetic gives more accurate representation because the n fractional bits give the fixed-point format much higher resolution. However, floating-point can represent a much wider range of numbers due to the variable exponent.

In addition to fixed- and floating-point representations, other possible numerical formats also exist. Kingsbury and Rayner report successful digital filter
implementations using logarithmic arithmetic [Ki71]. The 'Generalised fixed-point format' [Kim94a, Kim94b, Su94] has been developed to convert floating-point values to fixed-point for use in fixed-point digital signal processors.

### 3.3. The mechanism of coefficient representation error

The expression of coefficient representation error can be broken down into various components. By inspecting these constituents the mechanism for illustrating how coefficient representation error varies with respect to sampling frequency can be appreciated.

Consider the general digital filter \( G(z) \) with ideal coefficients:

\[
G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots} = \frac{A(z)}{B(z)}. \tag{3.15}
\]

Consider also the equivalent filter with coefficients which have been subjected to quantisation:

\[
H(z) = \frac{(a_0 + \alpha_i) + (a_1 + \alpha_1)z^{-1} + (a_2 + \alpha_2)z^{-2} + \ldots}{1 + (b_1 + \beta_i)z^{-1} + (b_2 + \beta_2)z^{-2} + \ldots} = \frac{A^*(z)}{B^*(z)}. \tag{3.16}
\]

where \( \alpha_i \) and \( \beta_i \) are the discrepancies caused by quantising the \( a_i \) and \( b_i \) coefficients respectively.

Using the definitions from [Fo91]:

\[
\alpha(z) = A^*(z) - A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots \tag{3.17}
\]

and

\[
\beta(z) = B^*(z) - B(z) = b_1 z^{-1} + b_2 z^{-2} + \ldots \tag{3.18}
\]
Equation (3.1) can be written in terms of $A(z)$, $B(z)$ and the quantised equivalents:

$$E_c(z) = u(z) \left[ \frac{A(z)}{B(z)} - \frac{A^*(z)}{B^*(z)} \right]$$  \hfill (3.19)

Cross-multiplying and dropping the "(z)" for notational convenience gives:

$$E_c(z) = u(z) \left[ \frac{AB^* - BA^*}{BB^*} \right]$$  \hfill (3.20)

Applying equations (3.17) and (3.18) to equation (3.20) and simplifying gives

$$E_c(z) = u(z) \left[ \frac{A \beta - B \alpha}{BB^*} \right]$$  \hfill (3.21)

Finally, taking the factor $1/B^+$ out of the bracketed expression results in

$$E_c(z) = u(z) \frac{1}{B^+} \left[ \frac{A \beta - \alpha}{B} \right]$$  \hfill (3.22)

This gives an expression with four terms ($\alpha$, $\beta$, $1/B^+$ and $A/B$) which assist in understanding the mechanism of how coefficient representation error varies for digital filters realised using the $z$-operator.

To demonstrate this, frequency domain analysis is carried out using a signal frequency of 1 radian per second to determine $\alpha$, $\beta$, $1/B^+$ and $A/B$. As an example, the digital filter of equation (3.3) is converted to the $z$-domain using the bilinear transform for a wide range of sampling frequencies and the coefficients are quantised to 12 bits.

### 3.3.1. Floating-point representation

The top left plot of Figure 3.8 illustrates how the magnitude of the term $A/B$ varies with sampling frequency. As the sampling rate is raised this term asymptotes towards a constant value. For slow sampling frequencies, the deviation from this constant value is quite marked. However this is consistent with algorithmic error analysis in that as the sampling rate is decreased, algorithmic error increases and behaviour of the digital filter deviates from that of its analogue counterpart.
The top right plot of Figure 3.8 illustrates how the magnitude of the $\alpha$ term varies as the sampling rate is changed. As the sampling rate is increased the size of the numerator coefficients in the digital filter diminish. Clearly as these coefficients become smaller then so do the quantised equivalents. Section 3.2 demonstrated that floating-point arithmetic has the capacity to represent small values with a high degree of accuracy. Therefore the discrepancy between the ideal and practical values decreases as the sampling rate is raised. However some values of $T$ lead to digital filters with practical coefficients which are close to the ideal ones, and other values of $T$ produce practical filter coefficients which deviate significantly from the ideal; hence the random fluctuations.

The bottom left curve of Figure 3.8 demonstrates how the magnitude of $\beta$ varies with changes of sampling rate. This indicates that $\beta$ varies in a random manner with increases of the sampling rate up to a point; after this point, the magnitude of $\beta$ decreases. As the sampling rate is increased, the poles of a digital filter migrate toward the point $z = 1$ in the $z$-plane and consequently the denominator of the digital transfer function approaches $z^2 - 2z + 1$. As the coefficients of the denominator retain the same order of magnitude over the range of sampling frequencies then the variation of $\beta$ is purely random. As the denominator of the ideal transfer function approaches $z^2 - 2z + 1$, coefficients $b_1$ and $b_2$ will eventually be forced by quantisation to take the values -2 and +1 respectively. Once this has happened, any further increases in sampling frequency will cause the ideal coefficients to approach the values -2 and +1 and therefore the discrepancy between the practical and ideal coefficients decreases - hence $\beta$ decreases uniformly.

The bottom right plot of Figure 3.8 shows how the quantity $1/B^+$ varies as the sampling interval is raised. The denominator of the ideal transfer function, $B(z)$, approaches $z^2 - 2z + 1$ as the sampling rate is increased. The magnitude of this quantity, when frequency analysis is performed, decreases as the sampling rate is raised. This leads to the trend that $1/B$ increases with increases in sampling frequency. When the magnitude of $B(z)$ is small the value of $\beta$ becomes significant and hence the variation
variation of $1/B^+$ is governed by the variation of $\beta$ for high sampling rates.

From equation (3.22), the coefficient representation error is proportional to $1/B^+$ and it therefore increases as the sampling rate is increased. As $\alpha$ decreases and $\beta$ fluctuates randomly for faster sampling rates, then the fluctuations of coefficient representation error are largely dependent on the accuracy by which the denominator coefficients are represented. Figure 3.9, which shows how coefficient representation error varies with sampling frequency for floating-point arithmetic, confirms this observation.

### 3.3.2. Fixed-Point Representation

The top left section of figure 3.10 shows how the quantity $A/B$ varies with sampling frequency. This is independent of quantisation and the variation of this quantity relates to algorithmic error.

The top right plot in Figure 3.10 shows how $\alpha$ varies with sampling frequency when fixed-point representations is used. As the sampling rate is raised the value of $\alpha$ tends to vary randomly up to a point then it diminishes with further increases in the sampling rate. From Section 3.2.1, the smallest number that a fixed-point representation can handle is $2^{-n}$, where $n$ is the number of fractional bits. The deviation of the practical coefficients from their ideal values will vary between $+/- 2^{-(n+1)}$, hence the variation of $\alpha$ appears initially to be purely random. For faster sampling rates the magnitudes of the $a_i$ coefficients (and their quantised equivalents) become smaller. A limit is eventually reached where the coefficients become so small that they are represented by zero, that is $a_i + \alpha_i = 0$. As the sampling rate is increased past this limit the values of the ideal $a_i$ coefficients approach zero such that the magnitude of $\alpha$ is equal to the magnitude of the diminishing coefficient.

The bottom left and bottom right graphs of Figure 3.10 respectively show how $\beta$ and $1/B^+$ vary with increases in sampling frequency. The trends in variations for these quantities reflect those for floating-point representation.
Figure 3.11 shows how coefficient representation error varies with sampling frequency when fixed-point arithmetic is used. The error increases in magnitude as the sampling rate is raised, albeit randomly. This observation is consistent with equation (3.22) and the results of Figure 3.10.

3.4. Coefficient Sensitivity

The quantised coefficients of a practical digital filter will in most circumstances differ from their ideal equivalents. This results in the positions of the poles and zeros in the z-plane of the practical digital filter deviating from the ideal. The response of the practical digital controller will therefore be different from that intended. By examining how deviations in the coefficients influence the behaviour of the practical digital filter, it is possible to determine coefficient wordlengths so a required filter accuracy can be achieved [Fo91, Go92, Go93].

It is a well known fact that digital filters of high order are prone to coefficient sensitivity problems. These problems can be reduced by implementing the digital transfer function as first- or second-order cascaded or paralleled subsections.

One technique of dealing with coefficient sensitivity is to investigate how the positions of the singularities vary with changes in the coefficients due to quantisation [Ka81, Ph90].

A more useful measure is to relate the changes in coefficient values of a digital filter to the equivalent continuous filter from which the discrete was derived [Fo91, Go92, Go93]. Suppose the second-order digital filter

$$G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (3.23)$$

is derived from the continuous equivalent, $F(s)$, where
Coefficient Representation Error

\[ F(s) = \frac{n_2 s^2 + n_1 s + n_0}{m_2 s^2 + m_1 s + 1}. \]  

(3.24)

This method determines sensitivity by the use of a matrix of sensitivity factors. For a general second-order filter this is expressed as

\[
\begin{bmatrix}
\frac{\partial n_0}{n_0} \\
\frac{\partial n_1}{n_1} \\
\frac{\partial n_2}{n_2} \\
\frac{\partial m_1}{m_1} \\
\frac{\partial m_2}{m_2}
\end{bmatrix}
= [S]
\begin{bmatrix}
\frac{\partial a_0}{a_0} \\
\frac{\partial a_1}{a_1} \\
\frac{\partial a_2}{a_2} \\
\frac{\partial b_1}{b_1} \\
\frac{\partial b_2}{b_2}
\end{bmatrix}
\]  

(3.25)

where

\[
[S] = \begin{bmatrix}
S_{n_0 a_0} & S_{n_0 a_1} & S_{n_0 a_2} & S_{n_0 b_1} & S_{n_0 b_2} \\
S_{n_1 a_0} & S_{n_1 a_1} & S_{n_1 a_2} & S_{n_1 b_1} & S_{n_1 b_2} \\
S_{n_2 a_0} & S_{n_2 a_1} & S_{n_2 a_2} & S_{n_2 b_1} & S_{n_2 b_2} \\
S_{m_1 a_0} & S_{m_1 a_1} & S_{m_1 a_2} & S_{m_1 b_1} & S_{m_1 b_2} \\
S_{m_2 a_0} & S_{m_2 a_1} & S_{m_2 a_2} & S_{m_2 b_1} & S_{m_2 b_2}
\end{bmatrix}
\]  

(3.26)

The elements of \([S]\) relate fractional changes in the discrete coefficients to fractional changes in the analogue coefficients. For example, the first element, or sensitivity factor, in \([S]\) relates fractional changes in the coefficient \(a_0\) to fractional changes in \(n_0\) where

\[ s_{n_0 a_0} = \frac{\partial n_0}{\partial a_0} \cdot \frac{a_0}{n_0}. \]  

(3.27)
Using the bilinear transform to relate $G(z)$ to $F(s)$, the sensitivity factors in $[S]$ can be expressed in terms of the discrete coefficients, viz

$$[S] = \begin{bmatrix}
\frac{a_0}{a_0 + a_1 + a_2} & \frac{a_1}{a_0 + a_1 + a_2} & \frac{a_2}{a_0 + a_1 + a_2} & -\frac{b_1}{1 + b_1 + b_2} & -\frac{b_2}{1 + b_1 + b_2} \\
\frac{a_0}{a_0 - a_2} & 0 & \frac{-a_2}{a_0 - a_2} & -\frac{b_1}{1 + b_1 + b_2} & -\frac{b_2}{1 + b_1 + b_2} \\
\frac{a_0}{a_0 - a_1 + a_2} & \frac{-a_1}{a_0 - a_1 + a_2} & \frac{a_2}{a_0 - a_1 + a_2} & -\frac{b_1}{1 + b_1 + b_2} & -\frac{b_2}{b_2(1 + b_1)} \\
0 & 0 & 0 & -\frac{-b_1}{1 + b_1 + b_2} & \frac{-b_2(2 + b_2)}{(1 - b_1)(1 + b_1 + b_2)} \\
0 & 0 & 0 & \frac{-2b_1(1 + b_2)}{(1 - b_1 + b_2)(1 + b_1 + b_2)} & \frac{2b_2}{(1 - b_1 + b_2)(1 + b_1 + b_2)}
\end{bmatrix}$$

If a sensitivity factor takes a value around 1, then it defines 'normal sensitivity'. That means a 1% change in the discrete filter coefficient will cause a change of around 1% in an equivalent continuous coefficient. A sensitivity factor which is significantly greater than 1 indicates 'high sensitivity' whilst a sensitivity factor significantly less than 1 indicates 'low sensitivity'.

The calculated sensitivity factors may be used to determine the required discrete coefficient accuracy requirements. This can be achieved by dividing the accuracy required in an equivalent continuous parameter by the root of the sum of the squares (rss) of the sensitivity factors of that parameter [Fo91].

Figure 3.12 shows how the rss values of the sensitivity factors, for the left-hand parameters in equation (3.25), vary as the sampling frequency is increased. This is carried out for the transfer function of equation (3.3). For various values of $T$, this continuous transfer function is transformed into the z-domain using the bilinear transform and sensitivity analysis is performed. Parameter $n_2$, which is zero in equation (3.3), has been omitted from the sensitivity analysis because any discrete coefficient variations causing this to change would represent infinitely high sensitivity. From
Figure 3.12, the rss of the sensitivity factors for all the continuous parameters is approximately equal. However the main point of interest is that coefficient sensitivity increases as the sampling rate is raised. This observation is consistent with the fact that coefficient representation error increases with faster sampling rates.

This analysis has shown how sensitivity factors can be calculated in terms of the coefficients of a second-order filter. It is also possible (and often more useful [Fo91]) to determine the sensitivity factors in terms of other continuous filter parameters such as the d.c. gain or the damping ratio. This could minimise the required computation as a full sensitivity analysis may not always be required.

3.5. Quantised-coefficient algorithmic error

Algorithmic error and coefficient representation error have so far been considered individually. In practical situations it is impossible to analyze these quantities in isolation because a practical digital filter will always have quantised coefficients. It is therefore convenient to define a new quantity which combines both these quantities - this will be termed 'quantised-coefficient algorithmic error' and denoted $E_d$ as this is a deterministic quantity.

Quantised-coefficient algorithmic error is calculated by evaluating, at each sampling instant, the difference between the outputs of a continuous filter and an equivalent digital filter with quantised coefficients. The z-transform of this error may be calculated, for an input $u(t)$ as

$$E_d(z) = Z\{u(s)F(s)\} - u(z)H(z)$$

(3.29)

where $F(s)$ is the continuous transfer function and $H(z)$ is the discrete transfer function with quantised coefficients. For residual frequency-domain analysis, assuming a unit sinusoid input of frequency $\omega$, the peak value of the error is given by
Quantised-coefficient algorithmic error is evaluated independently of any stochastic errors, such as ADC quantisation error or multiple-word truncation error.

Figure 3.13 shows how peak algorithmic error, coefficient representation error and quantised-coefficient algorithmic error vary as the sampling frequency is raised for a fixed unit input sinusoid of \( \omega = 0.1 \) radians per second. The analogue transfer function of equation (3.3) is converted to the z-plane using the bilinear transform and the coefficients are specified to 24 fractional bits. As the sampling rate is increased, the magnitude of algorithmic error decreases while that of coefficient representation error increases with random fluctuations. \( E_d \) is governed mainly by algorithmic error at 'low' sampling rates, and by coefficient representation error at 'high' sampling rates. At 'intermediate' sampling rates where algorithmic error and coefficient representation error are of the same order of magnitude, \( E_d \) is at its lowest. It therefore makes sense to sample at an intermediate sampling rate, and not to sample too quickly or too slowly.

Figure 3.14 demonstrates how the three quantities vary with signal frequency when the digital filter is sampling one hundred times per second. The plots of algorithmic error and coefficient representation error vary as expected. The plot of quantised-coefficient representation error consists of the vector sum of algorithmic error and coefficient representation error. An extremely interesting observation can be made for this particular example: when the algorithmic error and coefficient representation error are of similar orders of magnitude, the total error is smaller than the sums of the other two errors. This is because the residual algorithmic error and coefficient representation error can be represented as complex values. In this example, these error forms are out of phase with each other and a certain amount of error cancellation takes place. Not all digital filters produce errors which cancel, but those that do could yield high performance filters.
3.6. Summary

This chapter has presented an overview of coefficient representation error - the error generated when the coefficients of a practical digital filter are quantised. The variations of error have been examined with respect to sampling rate and signal frequency. For digital filters realised using the z-operator, coefficient representation error becomes of increasing concern as the sampling frequency is raised.

Numbers can be stored in computers using fixed-point and floating-point arithmetic. Other numerical formats exist but were not considered at much depth. A comparison between fixed-point and floating-point representations was carried out to demonstrate what number ranges each representation particularly suits.

The mechanism of coefficient representation error was considered to investigate how the error varies as the sampling rate is changed. This was achieved by splitting coefficient representation error into four sub-components and by examining the variations of these sub-components for increasing sampling frequencies.

Coefficient sensitivity was the next issue to be addressed. This relates fractional changes in the digital filter coefficients to fractional changes in the parameters of an equivalent continuous filter. Coefficient sensitivity increases as the sampling rate is increased, causing the poles of the discrete filter to move increasing closer to the +1 point on the unit circle in the z-plane. The method of examining coefficient sensitivity is useful in that it provides a useful basis for determining discrete coefficient wordlength requirements.

Finally, algorithmic error and coefficient representation error can be combined to give a quantity called 'quantised-coefficient algorithmic error'. By choosing a sampling frequency where this error is low, the performance of the resulting digital filter could be improved.
Figure 3.1. Model depicting coefficient representation error generation.

Figure 3.2. Variation of coefficient representation error during the residual state.
Figure 3.3. Variation of peak coefficient representation error against signal frequency.

Figure 3.4. Frequency response (magnitude) of the continuous transfer function of equation (3.3)
Coefficient Representation Error

Figure 3.5. Variation of coefficient representation error with sampling rate and signal frequency.

Figure 3.6. Basic fixed-point representation

Figure 3.7. Basic floating-point representation.
Figure 3.8. Variation of the components of coefficient representation error with sampling frequency for floating-point numbers.

Figure 3.9. Variation of coefficient representation error with sampling rate for floating-point arithmetic.
Figure 3.10. Variation of the components of coefficient representation error with sampling frequency for fixed-point numbers.

Figure 3.11. Variation of coefficient representation error with sampling rate for fixed-point arithmetic.
Figure 3.12. Variation of the r.s.s. of the coefficient sensitivity factors with sampling rate for the coefficients of equation (3.3)

Figure 3.13. Variation of algorithmic ($E_a$), coefficient representation ($E_c$) and quantised-coefficient algorithmic errors ($E_d$) with sampling rate.

Coefficient Representation Error
Variation of algorithmic ($E_a$), coefficient representation ($E_c$) and quantised-coefficient algorithmic ($E_d$) errors with signal frequency.

Figure 3.14. Variation of algorithmic ($E_a$), coefficient representation ($E_c$) and quantised-coefficient algorithmic ($E_d$) errors with signal frequency.
4.1 Introduction

This chapter examines the causes and effects of quantisation noise in digital filters. The generation of the noise and various analyses showing how it propagates through the filter will be presented.

Quantisation noise is generated as a consequence of fitting variables of infinite precision to finite precision wordlengths, or by the truncation or rounding of a multiple-wordlength product to a single- or lower-wordlength product. Analogue-to-digital converters are used to sample continuous signals and generate digital words whose magnitude is representative of the size of the input signal. The discrepancy between the signal represented by the digital value and the continuous signal is quantisation noise. Whenever a multiplication operation takes place in a digital filter algorithm, the product of say two single-wordlength values will produce a double-wordlength result. This invariably will be reduced to a single-wordlength value for the next iteration of the filter algorithm; the discrepancy between the single- and double-wordlength products results in quantisation noise.
4.1.1. Introduction to Rounding Error

Let $x$ be an unquantised variable. When $x$ is rounded to $N$ fractional bits, the rounded value is given by

$$
\hat{x}_r = \frac{\text{integer}\left( x2^N + 0.5 \right)}{2^N}
$$

(4.1)

In such a situation, the least significant bit (LSB) represents a value of $2^{-N}$.

Figure 4.1 illustrates the effects of rounding a continuous variable, $x$, to three fractional bits. The straight line indicates the unquantised variable, $x$, whilst the stepped function indicates the equivalent rounded values. From Figure 4.1, it is clear that the step size of the quantiser is 0.125 or $2^{-3}$. Figure 4.2 shows the discrepancy between the continuous value and the rounded value. The error for rounding, for this example lies between -0.0625 and +0.0625, i.e. between -0.5 LSB and +0.5 LSB, where the least significant bit represents 0.125.

4.1.2. Introduction to Truncation Error

Let $x$ be an exact value. When $x$ is truncated to $N$ fractional bits, the truncated value is given by

$$
\hat{x}_t = \frac{\text{integer}\left( x2^N \right)}{2^N}
$$

(4.2)

Again the least significant bit (LSB) represents a value of $2^{-N}$.

Figure 4.3 illustrates the effects of truncating a continuous variable, $x$, to three fractional bits. The straight line indicates the unquantised variable, $x$, whilst the stepped function indicates the equivalent truncated values. Figure 4.4 shows the discrepancy between the continuous value and the truncated value, which for this example lies between 0 and +0.125, i.e. between 0 and +1 LSB.
4.2. Methods of estimating the effects of roundoff noise

Suppose there exists a variable \( x(z) \) within a digital filter and \( H(z) \) is the transfer function relating \( x(z) \) to the output \( y(z) \). This system is depicted in Figure 4.5(a). Suppose that the variable \( x(z) \) is subject to quantisation such that the variable actually realised is \( x_q(z) \) where

\[
x_q(z) = x(z) + \epsilon(z)
\]  

(4.3)

and \( \epsilon(z) \) is the quantisation noise. From Figure 4.5(b), the output \( y_q(z) \) is due to the combined effects of \( x(z) \) and \( \epsilon(z) \).

From Figure 4.5(a) the ideal output \( y(z) \) is given by

\[
y(z) = x(z) H(z).
\]  

(4.4)

From Figure 4.5(b), the output \( y_q(z) \), which reflects the effects of quantisation noise is given by

\[
y_q(z) = x_q(z) H(z) = x(z)H(z) + \epsilon(z)H(z).
\]  

(4.5)

Let the noise, which has been transmitted through the system \( H(z) \) to the output be denoted \( y_s(z) \) then

\[
y_s(z) = y(z) - y_q(z) = \epsilon(z) H(z).
\]  

(4.6)

For systems which employ rounding, if the quantisation level of the wordlength representing the variable \( x(z) \) is \( q \) then the roundoff noise \( \epsilon(n) \) at each sampling instant \( n \) is bounded by

\[
-\frac{q}{2} \leq \epsilon(n) < +\frac{q}{2}.
\]  

(4.7)

Using the results of equations (4.6) and (4.7) the next three subsections review three
techniques by which the error of the digital filter is affected by quantisation noise. The first part details an overall worst-case error bound whilst the second part looks at a steady-state worst-case. The final subsection investigates a stochastic technique for calculating the average error.

4.2.1. Worst-Case Error Bound

Bertram [Be58] devised a method of bounding the error caused by quantisation noise based upon the pessimistic view that the roundoff noise causes the maximum possible harm to the output of the digital filter.

Writing equation (4.6) in the time-domain for the nth sampling instant gives

\[ y_q(n) = \sum_{k=0}^{n} h(k)e(n-k) \]  

(4.8)

where \( h(k) \) is the kth sample of the impulse response of \( H(z) \). In the case of roundoff noise, from equation (4.6) the magnitude of \( e \) is bounded by \( q/2 \). Taking the magnitudes of both sides of equation (4.8) leads to the inequality

\[ |y_q(n)| \leq | \sum_{k=0}^{n} h(k)e(n-k) | \]

\[ \leq \sum_{k=0}^{n} | h(k)e(n-k) | \]

(4.9)

\[ \leq \sum_{k=0}^{n} | h(k) || e(n-k) |. \]

The summation of expression (4.9) increases as the number of terms is increased. Also, from equation (4.6) it has been established that the maximum value of \( e \) is \( q/2 \). Therefore the error seen at the output of the digital filter due to roundoff noise is bounded by

\[ | y_q | \leq \sum_{k=0}^{\infty} | h(k) | \frac{q}{2}. \]  

(4.10)
However, this result is an extremely pessimistic estimation of the overall noise at the output of a digital filter.

Suppose there are $K$ sources of quantisation noise $q_1, q_2, \ldots, q_K$, in the digital filter whose transfer functions relating the output to the noise source are $H_1(z), H_2(z), \ldots, H_K(z)$ respectively, then the total error [Fr90] is bounded by

$$|y_q| \leq \left\{ \sum_{k=0}^{\infty} |h_1(k)| \frac{q_1}{2} + \sum_{k=0}^{\infty} |h_2(k)| \frac{q_2}{2} + \ldots + \sum_{k=0}^{\infty} |h_K(k)| \frac{q_K}{2} \right\}$$

(4.11)

4.2.2. Steady-State Worst-Case Error Bound

Slaughter [Sl64] formulated an alternative approach to estimating the error due to quantisation noise. This was based on the fact that roundoff noise causes transient errors of no concern and all variables become constant in the steady state. For this analysis the maximum steady-state error due to a constant maximum quantisation noise $q/2$ is determined.

From equation (4.8) at steady-state

$$y_{qss}(\infty) = \sum_{k=0}^{\infty} h(k)e_{ss}$$

(4.12)

where $e_{ss}$ is the worst-case steady-state value of the quantisation noise. So the worst-case steady-state error is found by taking the magnitude of this signal. Therefore

$$|y_{qss}(\infty)| \leq \left| \sum_{k=0}^{\infty} h(k) \frac{q}{2} \right|$$

(4.13)

Another way of looking at steady-state analysis is to recall that at steady state, the shift operator $z$ is equal to 1, so

$$|y_{qss}(\infty)| \leq |H(1)| \frac{q}{2}$$
Quantisation noise analysis

For K sources of quantisation noise $q_1$, $q_2$, ..., $q_K$ whose transfer functions relating the output to the noise source are $H_1(z)$, $H_2(z)$, ..., $H_K(z)$ respectively, then the total error is bounded by

$$|y_{eq}(\infty)| \leq \left\{ |H_1(1)| \frac{q_1}{2} + |H_2(1)| \frac{q_2}{2} + ... + |H_K(1)| \frac{q_k}{2} \right\}$$

(4.15)

Unfortunately the assumption of constant quantisation noise of $q/2$ does not always hold in the steady state. Even though this is not a bound on the error it provides a useful estimation of the magnitude of the error [Fr90]. Despite this it provides an expression that is easier to calculate than that of Bertram without being as pessimistic.

4.2.3. R.M.S. Estimate

The previous two sections have highlighted techniques to bound the magnitude of the error caused by quantisation noise. This section presents analysis which provides an average estimate for the error caused by quantisation noise. This technique features in many control texts, such as [Fr90] and [Ka81] and research papers, such as [Li71].

The analysis carried out in this section treats the noise introduced by the quantisation process as white noise. However this assumption is not strictly valid if the input to the quantisation process is deterministic, such as a sinewave or a square wave, for the noise spectrum will contain large spikes instead of being flat [Fr90]. For the purpose of this analysis, the assumption that the quantisation noise has a flat frequency spectrum will be observed.

Considering the case of round-off noise, Figure 4.6 illustrates the probability density function of the noise. From equation (4.7) it has already been established that the magnitude of the quantisation noise does not exceed $q/2$.

From probability theory, it is a well known fact [Ba89] that
Quantisation noise analysis

\[ \int_{-\infty}^{\infty} p(x) \, dx = 1 \]  \hspace{1cm} (4.16)

and therefore from the white noise model it is clear therefore that

\[
p(x) = \begin{cases} 
\frac{1}{q} & (|x| \leq \frac{q}{2}) \\
0 & (|x| > \frac{q}{2}).
\end{cases}
\]  \hspace{1cm} (4.17)

By inspection of Figure 4.6 it can be observed that the mean of the white noise is zero; the variance of the white noise \( \sigma_x^2 \) is found from

\[ \sigma_x^2 = \int_{-\frac{q}{2}}^{\frac{q}{2}} x^2 p(x) \, dx = \frac{1}{q} \int_{-\frac{q}{2}}^{\frac{q}{2}} x^2 \, dx = \frac{q^2}{12}. \]  \hspace{1cm} (4.18)

Assuming the roundoff noise \( \varepsilon(n) \) is based on a white noise model, then its autocorrelation function is given by [Fo89], [Fr90]

\[ R_\varepsilon(n) = E(\varepsilon(k)\varepsilon(k+n)) = \begin{cases} 
\frac{q^2}{12} & (n=0) \\
0 & (n \neq 0).
\end{cases} \]  \hspace{1cm} (4.19)

Using transform methods [Fr90] the equation relating the spectrum of the output error, \( S_y \), to the quantisation noise is

\[ S_y(z) = H(z) H(z^{-1}) S_\varepsilon(z) \]  \hspace{1cm} (4.20)

where \( H(z) \) is the transfer function relating the output to the noise source.

The spectrum of the input noise is found by taking the z-transform of its autocorrelation function. That is
From equations (4.20) and (4.21) the output spectrum is therefore given by

\[ S_y(z) = H(z) H(z^{-1}) \frac{q^2}{12}. \]  

(4.22)

Applying spectral analysis, the autocorrelation of the output noise is found by taking the inverse z-transform of the output spectrum, so

\[ R_y(n) = Z^{-1}\{S_y(z)\}. \]

(4.23)

The variance of the output noise \( \sigma_y^2 \) equals the autocorrelation function of \( y_q \) at \( n=0 \). Thus

\[ \sigma_y^2 = R_y(0) \]

\[ = \frac{1}{2\pi j} \oint_{|z|=1} S_y(z) \frac{dz}{z} \]

(4.24)

\[ = \frac{1}{2\pi j} \oint_{|z|=1} \{H(z) H(z^{-1}) z^{-1} dz\} \frac{q^2}{12}. \]

The result of equation (4.24) may be evaluated using Cauchy's residue formula. The r.m.s. value of the quantisation error is

\[ \varepsilon_q = \sqrt{\sigma_y^2}. \]  

(4.25)

4.2.4. A Comparison Of The Three Error Estimates

Consider a digital filter (with sampling interval \( T=0.5 \) seconds) where the transfer function relating the filter output to a random noise source, \( H(z) \), is
Quantisation noise analysis

\[ H(z) = \frac{15z - 35/3}{z - 1/3}. \]  (4.26)

Considering first, Bertram's worst-case bound on error, \( H(z) \) can be written as

\[ H(z) = 15 - \frac{20/3}{z - 1/3}. \]  (4.27)

From equation (4.11) the absolute worst case error that a rounding-noise source can generate can be calculated as follows:

\[
|y_q| \leq \left\{ \sum_{k=0}^{\infty} 15 \times 0^k \left| \frac{q}{2} \right| + \sum_{k=0}^{\infty} -\frac{20}{3} \times \left( \frac{1}{3} \right)^{k+1} \left| \frac{q}{2} \right| \right\} 
\leq \frac{q}{2} \left( 15 + 10 \right) 
\leq 12.5q.
\]  (4.28)

Considering now, Slaughter's steady-state worst case error bound. From equation (4.14)

\[
|y_q| \leq |H(1)| \frac{q}{2} 
\leq \frac{15 - 35/3}{1 - 1/3} \frac{q}{2} 
\leq 2.5q.
\]  (4.29)

Considering finally, the r.m.s. case. The product \( H(z)H(z^{-1})z^{-1} \) is calculated as

\[
H(z)H(z^{-1})z^{-1} = \frac{15z - 35/3}{z - 1/3} \cdot \frac{15z^{-1} - 35/3}{z^{-1} - 1/3} \cdot \frac{1}{z} 
= \frac{15z - 35/3}{z - 1/3} \cdot \frac{15 - 35/3z}{1 - 1/3z} \cdot \frac{1}{z} 
= \frac{250}{z - 3} - \frac{250}{z - 1/3} + \frac{525}{z}.
\]  (4.30)

Using equation (4.24) and applying Cauchy's residue formula
Quantisation noise analysis

\[ \sigma_q^2 = \frac{1}{2\pi j} \oint_{|z|=1} \{ H(z) H(z^{-1}) z^{-1}dz \} \frac{q^2}{12} \]

\[ = \frac{q^2}{12} \cdot \frac{1}{2\pi j} \cdot (0 - 250(2\pi j) + 525(2\pi j)) \]

\[ = \frac{275q^2}{12}. \tag{4.31} \]

Finally from equation (4.25) the r.m.s. error is given by

\[ e_t = \sqrt{\sigma_q^2} = q \sqrt{\frac{275}{12}} = 4.8q. \tag{4.32} \]

This example simply illustrates the computation processes involved in evaluating the three error estimates. From equations (4.28) and (4.32), Bertram’s method produces estimations which far exceed that of the r.m.s. case indicating the pessimism of the worst-case bound. By comparison of equations (4.29) and (4.32), despite the value given by Slaughter’s method differing from the r.m.s. value, it still provides an estimate for the error which is useful at low sampling rates and requires minimal computation.

For digital filters emulated using the bilinear transform from

\[ F(s) = \frac{1}{s^2 + 4s + 3}, \tag{4.33} \]

with sampling rates varying from 10 to 1000 samples per second, Figure 4.7 shows the normalised quantisation error estimations versus sampling frequency for the worst-case, steady-state worst-case and the r.m.s noise analyses. This graph also reflects the pessimism of Bertram’s method but shows that Slaughter’s method is not as pessimistic. However, as the sampling frequency is raised, the r.m.s. value of the error decreases and the discrepancy between the r.m.s. error and the error bounds becomes quite marked, especially at high sampling rates.
4.3. Numerical methods for evaluating the contour integral

The most reliable method of estimating the quantisation error is to use the r.m.s. technique. This section reviews some of the techniques that can be used to evaluate the contour integral of equation (4.24) in order that the average error may be determined.

4.3.1. Cauchy's Residue Theorem

The formal way to evaluate the contour integral of equation (4.24) would be to apply Cauchy's residue theorem [St91], [Ch84] which states that if the complex function \( f(z) \) is regular at every point within and on a closed curve, \( c \), then

\[
\oint_{c} f(z) \, dz = 0. \tag{4.34}
\]

However, when the function \( f(z) \) contains singularities (poles), which is the case for recursive digital filters equation \( f(z) \) ceases to be regular. Suppose \( f(z) \) is of the form

\[
f(z) = \frac{1}{(z - a)^n}
\]

it can be shown [St91] that

\[
\oint_{c} \frac{1}{(z - a)^n} \, dz = \begin{cases} 0 & \text{if } n \neq 1 \\ 2\pi i & \text{if } n = 1 \text{ and } c \text{ encloses } z = a. \end{cases} \tag{4.35}
\]

This can be applied to the following general example where

\[
f_i(z) = \frac{A}{z - a} + \frac{B}{z - b}. \tag{4.36}
\]

Suppose this function requires integration over the contour \( c \), and both points \( a \) and \( b \) lie outside the contour then
Quantisation noise analysis

\[ \oint_{c} f_t(z) \, dz = 0. \]  
\[ (4.37) \]

Suppose point a is situated inside the contour and b outside then

\[ \oint_{c} f_t(z) \, dz = (2\pi j)A. \]  
\[ (4.38) \]

Suppose this time, the situation is reversed; point a lies outside the contour and b lies within the contour then

\[ \oint_{c} f_t(z) \, dz = (2\pi j)B. \]  
\[ (4.39) \]

For the case where both points a and b lie inside the contour c

\[ \oint_{c} f_t(z) \, dz = (2\pi j)A + (2\pi j)B. \]  
\[ (4.40) \]

To evaluate the contour integral of equation (4.24), the first stage is to form the product \( H(z)H(z^{-1})z^{-1} \) and express this as partial fractions. The contour integration can be readily evaluated by applying equation (4.35) to each of the terms. For digital filters realised using the shift operator, the contour, c, is the unit circle which is centred at the origin of the z-plane.

This can be readily calculated using MATLAB. The general transfer function

\[ H(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}}{1 + b_1 z^{-1} + \ldots + b_m z^{-m}}. \]  
\[ (4.41) \]

can be represented by the row vectors

\[ A = [a_0 \ a_1 \ \ldots \ a_n] \quad \text{and} \quad B = [1 \ b_1 \ \ldots \ b_m] \]  
\[ (4.42) \]

respectively. The product
Quantisation noise analysis

\[
\frac{H(z)H(z^{-1})}{z} = \frac{a_0 + a_1z^{-1} + \ldots + a_n}{z^n + b_1z^{n-1} + \ldots + b_n} \frac{1}{1 + b_1z^{-1} + \ldots + b_nz^{-n}}
\]  

(4.43)

can be determined in MATLAB using:

```
NUM=conv(A, fliplr(A));
DEN=conv(conv(B,fliplr(B)),[1 0]);
```

where NUM and DEN are respectively the row vectors which contain the coefficients of the numerator and denominator of equation (4.43).

The residues (R), poles (P) and constant term (K) for the partial fraction expansion of equation (4.42) can be determined in MATLAB using

```
[R, P, K]=residue(NUM, DEN);
```

Given the values of R, P and K, the contour integral of equation (4.24) can readily be evaluated by examining the values and the multiplicities of the poles to find out which terms lie inside the unit circle; application of equation (4.35) will yield the solution.

4.3.2. Jury’s Numerical Method

For a general \(n\)th-order transfer function \(H(z)\) where

\[
H(z) = \frac{a_0 + a_1z^{-1} + a_2z^{-2} + \ldots + a_nz^{-n}}{1 + b_1z^{-1} + b_2z^{-2} + \ldots + b_nz^{-n}}
\]  

(4.44)

Jury presented a simple method [Ju65], [Ju66] that can be used to numerically evaluate the contour integral of equation (4.24). This is presented as

\[
I = \frac{|\Omega_1|}{|\Omega|} = \frac{1}{2\pi j} \oint_{|z|=1} H(z)H(z^{-1})z^{-1}dz
\]  

(4.45)

where \(|\Omega_1|\) and \(|\Omega|\) are respectively the determinants of \(\Omega_1\) and \(\Omega\).
Quantisation noise analysis

where

\[ \Omega = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & \ldots & b_n \\ b_1 & b_0 + b_2 & b_1 + b_3 & b_2 + b_4 & \ldots & b_{n-1} \\ b_2 & b_3 & b_0 + b_4 & b_3 + b_5 & \ldots & b_{n-2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_n & 0 & 0 & 0 & \ldots & b_0 \end{bmatrix} \quad (4.46) \]

(where \( b_0 = 1 \)) and the matrix \( \Omega_1 \) is formed from the matrix \( \Omega \) by replacing the first column by

\[ \begin{bmatrix} \sum_{i=0}^{n} a_i^2 & 2\sum a_ia_{i+1} & 2\sum a_ia_{i+2} & \ldots & 2\sum a_ia_{i,n-1} & 2a_n \end{bmatrix}^T \quad (4.47) \]

For a second order system,

\[ H(z) = \frac{a_0 + a_1z^{-1} + a_2z^{-2}}{1 + b_1z^{-1} + b_2z^{-2}} \quad (4.48) \]

the required matrices are therefore

\[ \Omega = \begin{bmatrix} b_0 & b_1 & b_2 \\ b_1 & b_0 + b_2 & b_1 \\ b_2 & 0 & b_0 \end{bmatrix} \quad (4.49) \]

and

\[ \Omega_1 = \begin{bmatrix} a_0^2 + a_1^2 + a_2^2 & b_1 & b_2 \\ 2(a_0a_1 + a_1a_2) & b_1 + b_2 & b_1 \\ 2a_2a_2 & 0 & b_0 \end{bmatrix} \quad (4.50) \]
4.3.3. Åström's Numerical Method

In 1970, Åström et al produced an alternative numerical method which could be used to evaluate complex integrals [As70]. The authors (of which one was Jury) claimed that this method was the simplest yet obtained.

For the discrete first-order transfer function

\[ H(z) = \frac{a_0 + a_1 z^{-1}}{b_0 + b_1 z^{-1}} \]  

(4.51)

the contour integral of equation (4.24) can be evaluated using the formula

\[ \Phi_0 = \frac{1}{2\pi j} \oint_{|z|=1} H(z) H(z^{-1}) z^{-1} \, dz \]

(4.52)

\[ = \frac{1}{b_0} \left[ \left( \frac{a_0^0}{b_0^0} \right)^2 + \left( \frac{a_1}{b_0} \right)^2 \right] \]

where the coefficients \( b_0^0 \) and \( a_0^0 \) may respectively be determined using

\[ b_0^0 = \frac{(b_0)^2 - (b_1)^2}{a_0} \]  

(4.53)

and

\[ a_0^0 = \frac{b_0 a_0 - b_1 a_1}{b_0} \]  

(4.54)

For a second-order digital transfer function, this integral is evaluated using

\[ \Phi_0 = \left[ \frac{(a_0^0)^2}{b_0^0} + \frac{(a_1)^2}{b_1} + \frac{(a_2)^2}{b_0} \right] \]

(4.55)

where
Quantisation noise analysis

\[ b_1^1 = \frac{b_0b_1 - b_1b_2}{b_0} \]  \hspace{1cm} (4.56)

\[ a_0^1 = \frac{b_0a_0 - b_2a_2}{b_0} \]  \hspace{1cm} (4.57)

\[ b_0^0 = \frac{(b_0^1)^2 - (b_1^1)^2}{b_0^1} \]  \hspace{1cm} (4.58)

\[ a_0^0 = \frac{b_0^1a_0^1 - b_1^1a_1^1}{b_0^1} \]  \hspace{1cm} (4.59)

4.4. A Corrected Numerical Method For Evaluating The Contour Integral

The analysis of this section corrects the work of [Fo89] and extends it to third-order systems. It would be possible to extend the systems to higher-order but this seems unnecessary as most digital controllers are realised using first- or second-order sections; in rare cases third order systems may be implemented. Systems of higher-order are generally implemented as either a cascade or a parallel combination of first-order or second-order sub-sections.

4.4.1. Simple Poles

A digital transfer function whose general form is

\[ H(z) = \frac{a_0 + a_1z^{-1} + \ldots + a_nz^{-n}}{1 + b_1z^{-1} + \ldots + b_nz^{-n}} = \frac{a_0z^n + a_1z^{n-1} + \ldots + a_n}{z^n + b_1z^{n-1} + \ldots + b_n} \]  \hspace{1cm} (4.60)

may be expressed as the general partial fraction expansion
Quantisation noise analysis

\[
H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{r_2}{z - p_2} + \ldots + \frac{r_n}{z - p_n}
\]  \hspace{1cm} (4.61)

where the value of each \( p_i \) corresponds to the value of the pole. The impulse response sequence of equation (4.61) is

\[
h(k) = \{ r_0, (r_1 + r_2 + \ldots + r_n), (r_2 + r_3 + \ldots + r_n), (r_3 + r_4 + \ldots + r_n)^2, \ldots \} \] \hspace{1cm} (4.62)

or

\[
h(k) = r_00^k + r_1p_1^{k-1} + r_2p_2^{k-1} + \ldots + r_np_n^{k-1}. \] \hspace{1cm} (4.63)

This assumes that all the poles lie inside the unit circle such that the value of \( h(k) \) converges to zero as the sample number, \( k \), increases. Also from equation (4.62)

\[
rp_i^{k-1} \leq 0 \hspace{1cm} \forall k \leq 1 \] \hspace{1cm} (4.64)

Another technique for determining the r.m.s. error due to quantisation noise is to evaluate the infinite series for the square of \( h(k) \) [Ra75]. That is

\[
\sigma_y^2 = \sigma_x^2 \sum_{k=0}^{\infty} [h(k)]^2 \] \hspace{1cm} (4.65)

\[
= \sigma_x^2 \Phi_0
\]

where \( \Phi_0 \) is the autocorrelation of the output at the current time instant.

Using equation (4.63), \( \Phi_0 \) can be expressed as

\[
\Phi_0 = \sum_{k=0}^{\infty} \left[ r_00^k + \sum_{i=1}^{n} rp_i^{k-1} \right] \left[ r_00^k + \sum_{j=1}^{n} rp_j^{k-1} \right]
\]

\[
= \sum_{k=0}^{\infty} \left[ r_00^2k + r_00^k \sum_{j=1}^{n} rp_j^{k-1} + r_00^k \sum_{i=1}^{n} rp_i^{k-1} + \sum_{i=1}^{n} \sum_{j=1}^{n} rp_i^{k-1}rp_j^{k-1} \right] \] \hspace{1cm} (4.66)

\[
\Phi_0 = \frac{1}{2\pi j} \oint_{|z|=1} H(z)H(z^{-1}) \frac{dz}{z}
\]

However, from equations (4.63) and (4.64) it is apparent that
Quantisation noise analysis

\[ r_0^0 \sum_{k=0}^{\infty} r p_{k+1} = 0. \]  

(4.67)

It therefore follows that

\[ \Phi_0 = \sum_{k=0}^{\infty} \left[ r_0^2 s^{2k} + \sum_{i=1}^{n} \sum_{f=1}^{n} r f (p f p)_{k+1} \right] \]

\[ = r_0^2 + \sum_{i=1}^{n} \sum_{f=1}^{n} r f \left( \frac{1}{1 - p f} \right) \]

\[ = [R][Q][R]^T \]

(4.68)

after performing the infinite summations (see Appendix A). The [R] and [Q] matrices, from equation (4.68) are respectively

\[ [R] = [r_0 \ r_1 \ r_2 \ ... \ r_s] \]

(4.69)

and

\[ [Q] = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1 p_2} & \ldots & \frac{1}{1 - p_1 p_s} \\ 0 & \frac{1}{1 - p_2 p_1} & 1 & \frac{1}{1 - p_2^2} & \ldots & \frac{1}{1 - p_2 p_s} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \frac{1}{1 - p_s p_1} & \frac{1}{1 - p_s p_2} & \ldots & 1 & \frac{1}{1 - p_s^2} \end{bmatrix} \]

(4.70)

The equivalent [Q] matrix in [Fo89] differs from that of expression (4.70) only in that the zeros are replaced by ones. This discrepancy occurs due to an incorrect partial expansion of H(z) in [Fo89].

4.4.2. Second-Order Systems With Double Pole

Consider a second-order system which, when expanded into partial fractions, is
Quantisation noise analysis

\[ H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{r_2}{(z - p_1)^2} + \frac{r_3}{(z - p_1)^3}. \]  
(4.71)

The contour integral of equation (4.24) is numerically evaluated using equation (4.66); the \([R]\) and \([Q]\) matrices are respectively given by

\[
[R] = \begin{bmatrix}
  r_0 & r_1 & r_2 \end{bmatrix}
\]  
(4.72)

and

\[
[Q] = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{p_1}{1 - p_1^2} \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{p_1}{1 - p_1^2} \\
  0 & \frac{p_1}{1 - p_1^2} & \frac{p_2}{1 - p_1^2} & \frac{1 + p_1^2}{1 - p_1^2}
\end{bmatrix}
\]  
(4.73)

4.4.3. Third-Order Systems With Double Pole

Consider the general third-order transfer function having a double pole

\[ H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{r_2}{z - p_2} + \frac{r_3}{(z - p_1)^2}. \]  
(4.74)

The contour integral of equation (4.66) can now be evaluated using

\[
[R] = \begin{bmatrix}
  r_0 & r_1 & r_2 & r_3 \end{bmatrix} \text{ and } 
\]  
(4.75)

\[
[Q] = \begin{bmatrix}
  1 & 0 & 0 & 0 & 0 \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{p_1}{1 - p_1^2} \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{p_2}{1 - p_1^2} \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1 + p_2^2}{1 - p_1^2} \\
  0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1^2} & \frac{1 + p_2^2}{1 - p_1^2}
\end{bmatrix}
\]  
(4.76)
4.4.4. Third Order System With Triple Pole

Consider, finally a general third-order digital transfer function with a single pole located at the point \( z = p \):

\[
H(z) = r_0 + \frac{r_1}{z - p} + \frac{r_2}{(z - p)^2} + \frac{r_3}{(z - p)^3} \tag{4.77}
\]

The value of the contour integral, given by \([R][Q][R]^T\), may be found in the usual way with

\[
[R] = \begin{bmatrix} r_0 & r_1 & r_2 & r_3 \end{bmatrix} \tag{4.78}
\]

and

\[
[Q] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & p & p^2 \\ 0 & \frac{1}{1 - p^2} & \frac{p}{(1 - p^2)^2} & \frac{p^2}{(1 - p^2)^3} \\ 0 & \frac{p}{1 - p^2} & \frac{1 + p^2}{(1 - p^2)^3} & \frac{2p^3 + p}{(1 - p^2)^4} \\ 0 & \frac{p^2}{(1 - p^2)^3} & \frac{2p^3 + p}{(1 - p^2)^4} & \frac{1 + 4p^2 + p^4}{(1 - p^2)^5} \end{bmatrix} \tag{4.79}
\]

The derivations of the \([Q]\) matrices are laid out in Appendix B.

4.4.5. Transfer Functions Having Complex Poles

As the analysis carried out so far has been restricted to transfer functions of up to third order the analysis involving systems with complex poles will also be constrained to third-order transfer functions; systems of higher order can be treated as a catenation of lower order subsections. A second order transfer function with complex roots takes the general form

\[
H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{\bar{r}_1}{z - \bar{p}_1} \tag{4.80}
\]

whilst a third order system will be of the following form
Quantisation noise analysis

\[ H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{\overline{r_1}}{z - \overline{p_1}} + \frac{r_3}{z - p_3} \]  

(4.81)

where \( r_1 \) is a complex residue and \( p_1 \) is a complex pole; their complex conjugates also exist in equations (4.80) and (4.81). The third-order system also has an additional term containing a real residue and a simple real pole.

By inspection of equations (4.80) and (4.81) it is apparent that the multiplicity of both systems is 1 and therefore the method of computing the contour integral using the numerical method with equations (4.68) to (4.70) can readily be applied.

However, a degree of care needs to be undertaken in order to achieve correct results with this technique as the row matrix \([R]\) consists of complex elements. The MATLAB transpose operator transposes a complex matrix in the formal sense [Ma92]. That is the result of such a transpose will be the complex conjugate of the required result, the unconjugated transpose. The formal complex matrix transpose of the complex matrix \( R \) is known as the Hermitian transpose, denoted by \( R^H \) [No88], [Str88]. In order to avoid confusion, \( R^T \) will be defined as

\[ [R]^T \triangleq [\overline{R}]^H. \]  

(4.82)

4.5. Examples

This section presents some examples to demonstrate the effectiveness of the technique of evaluating the contour integral which was developed in the previous section. It also makes a comparison between the effectiveness of the new technique against the established techniques which were outlined in Section 4.3.

4.5.1. The Effectiveness Of The New Technique

The tests which follow in this subsection are performed on all types of third order digital filters.
Quantisation noise analysis

Considering firstly the simple case where all the poles are distinct. From the first order continuous transfer function

\[ F(s) = \frac{6}{(s+1)(s+2)(s+3)} \]  

(4.83)

using the bilinear transform with a sampling time of 0.5 seconds, the resulting digital transfer function is

\[ H(z) = 0.0286 + \frac{0.9600}{z - 0.6000} - \frac{1.3333}{z - 0.3333} + \frac{0.4898}{z - 0.1429}. \]  

(4.84)

Using equations (4.68), (4.69) and (4.70), the value of \( \Phi_0 \) is

\[ \Phi_0 = [R][Q][R]^T = 0.1429. \]  

(4.85)

Defining \( \Phi_d \) from equation (4.65) as a datum to compare the results of the new method it was found that

\[ \Phi_d = \sum_{k=0}^{100000} [h(k)]^2 = 0.1429. \]  

(4.86)

The upper limit of 100,000 is suitably large to ensure accurate results; the value of \( h(k) \) at \( k=100,000 \) is negligible.

To appreciate the practical significance of these results, the filter \( H(z) \) was realised in software and was subjected to an input of 100,000 samples of random noise. The noise, whose value varied between +0.5 and -0.5 had a measured variance of 0.0832. From equation (4.24), the theoretical value of the variance of the output noise is equal to the input variance multiplied by \( \Phi_0 \). For this example, the theoretical variance of the output noise was 0.0119 to 4 decimal places. The measured variance of the output noise was also found to be 0.0119 to 4 decimal places; the actual discrepancy being just 0.32%.

Consider now a third order case with a double pole at \( z = 0.8 \). An example of such a digital transfer function is
Applying equations (4.68), (4.75) and (4.76) it can be shown that $\Phi_0 = 5.7097$, which agrees with the datum result of $\Phi_d = 5.7097$.

Considering a digital transfer function with multiplicity three. An example of a such a system with a triple pole at $z = 0.9$ is

$$H(z) = 0.9 + \frac{0.1}{z - 0.9} + \frac{0.2}{z - 0.8} + \frac{0.3}{(z - 0.8)^2}. \quad (4.87)$$

Using equations (4.68), (4.78) and (4.79) the value of $\Phi_0$, was found to be 2030.0. This agrees with the datum value of $\Phi_d = 2030.0$.

Finally the following third order transfer function has a conjugate pair of complex poles:

$$H(z) = 0.9 + \frac{0.1 - j0.2}{z - (0.8 + j0.1)} + \frac{0.1 + j0.2}{z - (0.8 - j0.1)} + \frac{0.3}{z - 0.9}. \quad (4.89)$$

This system essentially has distinct poles so using equations (4.68) to (4.70) and observing the definition of the transposed R matrix as given in equation (4.82), the value of $\Phi_0$ was found to be 1.4927, agreeing with the actual value of $\Phi_d = 1.4927$.

4.5.2. A Comparison Of Methods Used To Compute The Contour Integral

It has already been demonstrated that the corrected method of evaluating the contour integral of equation (4.24) is an effective process. This subsection compares each of the first three techniques of section 4.3 and the corrected method to the value of a datum. The percentage inaccuracy is recorded such that a comparison of accuracy between the four methods can be carried out. The tests were performed using MATLAB.
Quantisation noise analysis

As a starting point, consider the continuous transfer function

\[ F(s) = \frac{10s + 10}{s^2 + s + 10}. \]  (4.90)

Using the bilinear transform and a sampling interval of \( T = 0.1 \) seconds, the equivalent digital transfer function can be emulated. Using

(a) Cauchy's method
(b) Jury's method
(c) Åström's method, and
(d) the new method

the percentage accuracy of the computation can be determined against a datum figure, this being the equivalent \( \Phi_d \) that was used in the previous analysis. (The upper limit of 100,000 for \( \Phi_d \) still ensures an extremely high level of accuracy.) This process will be repeated for other values of sampling interval (\( T = 0.01 \) seconds and \( T = 0.001 \) seconds) to build up a set of results in order that a fair comparison can be made.

The results of the computations are illustrated in Table 4.1.

For the continuous transfer function

\[ F(s) = \frac{s + 4}{s^2 + 3s + 2}. \]  (4.91)

which has purely real poles, Table 4.2 shows similar analysis for the four methods of evaluating the contour integral of equation (4.24) for the equivalent digital filter.

By inspection of both Tables 4.1 and 4.2, it is interesting to observe that the value of the contour integral, denoted as the datum value decreases as the sampling rate is raised. As the sampling rate of a digital filter is increased, the numerator coefficients become smaller (at the same time the poles move toward the point \( z = 1 \), in order to maintain the filter gain). As a consequence of diminishing numerator coefficients, partial fraction expansion techniques yield smaller residues, causing the value of the
contour integral to decrease, according to Cauchy's method.

On comparing the accuracies of the techniques carried out in MATLAB, Cauchy's method produces the greatest discrepancy. Suppose the transfer function $H(z)$ is $n^{th}$ order, then the order of the product $H(z)H(z^{-1})z^{-1}$ will be $(2n + 1)$. A partial fraction expansion can be carried out on the product, to determine the poles and residues, which involves the use of root finding algorithms. As root finding algorithms are usually performed as an iterative process, accuracy is not always guaranteed. Problems will also arise if the order of the polynomial is high or if the polynomial itself is ill-conditioned. These factors all contribute to the relatively poor accuracy achieved using Cauchy's residue theorem.

Greater accuracy can be achieved using either Jury's or Åström's method. For certain situations Jury's method gives the greater degree of accuracy, whilst for other situations Åström's method is superior. Jury's method is relatively simple to implement while Åström's requires several intermediate calculations.

The results demonstrate that the best means of determining the value of the contour integral in terms of accuracy comes from the corrected method that was discussed in Section 4.4. For all but one configuration highlighted in Tables 4.1 and 4.2 the new method clearly yields the most accurate results. As with the other three methods, as the sampling frequency is raised the discrepancy between the theoretical and calculated values increases. However the rate of increase in discrepancy with respect to sampling frequency is lower than the other three techniques. This makes the corrected technique reliable at high sampling rates where the other methods could exhibit significant inaccuracies.

4.6. Summary

This chapter has presented several methods by which the propagation of quantisation noise may be estimated. One of the simplest methods of computing the error is simply
Quantisation noise analysis

to compute a maximum bound for the error. Bertram produced a method of bounding the error which is excessively pessimistic. Slaughter later devised a means of determining the steady-state worst-case error bound; this being easier to calculate and more meaningful than that of Bertram.

A more useful measure is the technique of predicting the average value of the quantisation error, of which several existing techniques have been reviewed. A corrected technique has been developed in this Chapter which is relatively simple to apply provided that the poles and residues of the transfer function can be readily determined. It has been demonstrated that the accuracy of this technique is greater than any of the existing techniques investigated.

Figure 4.1. Demonstrating the effects of roundoff.
Quantisation noise analysis

Figure 4.2. Discrepancy caused by roundoff.

Figure 4.3. Demonstrating the effects of truncation.
Quantisation noise analysis

Figure 4.4. Discrepancy caused by truncation.

(a) Linear System

(b) Linear system with quantisation noise on the variable $x(z)$

Figure 4.5. Introduction of roundoff noise into a linear system.
Figure 4.6. P.D.F. model of roundoff noise.

Figure 4.7. Quantisation error estimates versus sampling frequency for a second-order filter, for the three methods.
### Table 4.1
Comparing numerical contour integration methods for a second-order lightly-damped transfer function.

<table>
<thead>
<tr>
<th>Method</th>
<th>T=0.1 seconds</th>
<th>T=0.01 seconds</th>
<th>T=0.001 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum Value</td>
<td>5.1395</td>
<td>0.5474</td>
<td>0.0550</td>
</tr>
<tr>
<td>Percentage Error</td>
<td>7.1372 × 10^{-12}</td>
<td>-1.9140 × 10^{-7}</td>
<td>0.0018</td>
</tr>
<tr>
<td>Cauchy's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>-1.0023 × 10^{-12}</td>
<td>-1.2607 × 10^{-9}</td>
<td>-2.6646 × 10^{-7}</td>
</tr>
<tr>
<td>Jury's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>-6.9125 × 10^{-14}</td>
<td>2.6266 × 10^{-11}</td>
<td>-3.8843 × 10^{-8}</td>
</tr>
<tr>
<td>Åström's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>-1.0369 × 10^{-13}</td>
<td>-3.3061 × 10^{-12}</td>
<td>-3.9141 × 10^{-11}</td>
</tr>
<tr>
<td>New Method</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.2
Comparing numerical contour integration methods for a second-order transfer function with real poles.

<table>
<thead>
<tr>
<th>Method</th>
<th>T=0.1 seconds</th>
<th>T=0.01 seconds</th>
<th>T=0.001 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum Value</td>
<td>0.1472</td>
<td>0.0150</td>
<td>0.0015</td>
</tr>
<tr>
<td>Percentage Error</td>
<td>-2.3517 × 10^{-10}</td>
<td>-3.1194 × 10^{-7}</td>
<td>0.0214</td>
</tr>
<tr>
<td>Cauchy's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>6.9773 × 10^{-13}</td>
<td>-1.5108 × 10^{-9}</td>
<td>1.3823 × 10^{-6}</td>
</tr>
<tr>
<td>Jury's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>2.0932 × 10^{-12}</td>
<td>-2.3014 × 10^{-9}</td>
<td>-1.9821 × 10^{-6}</td>
</tr>
<tr>
<td>Åström's method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage Error</td>
<td>-2.8286 × 10^{-13}</td>
<td>4.0777 × 10^{-12}</td>
<td>-7.5748 × 10^{-11}</td>
</tr>
<tr>
<td>New Method</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER FIVE
A COMPARISON OF ALTERNATIVE DIGITAL FILTER OPERATORS

5.1 Introduction

Digital filters realised using the shift-operator, which is widely used in digital filtering applications, have always been prone to errors due to finite wordlength effects particularly at high sampling frequencies. Several alternative operators have been proposed as an attempt to remedy these problems, of which some will be discussed in this chapter.

In 1975, Agarwal and Burrus [Ag75] developed a new operator defined as

\[ \xi = z - 1 \]  

(5.1)

to counter the problems associated with fast sampling rates where the poles of a digital filter, realised using the shift operator, migrate towards the point \( z = 1 \) in the \( z \)-plane causing coefficient sensitivity effects.

Bolton [Bo80] devised an alternative operator called the Y-operator defined as

\[ y = \frac{(1 - z^{-1})}{T} \]  

(5.2)
A comparison of alternative digital filter operators

which gives digital differentiation. For high sampling rates, the \( y \)-operator closely approximates the Laplace operator, \( s \). It is claimed that the inverse filter element, \( y^{-1} \) gives improved coefficient sensitivity but accurate adjustments must be made to counteract the effects of sampling.

Also to overcome the problems inherent in the use of the shift operator, \( z \), in digital filter design, two alternatives have been proposed, both known as delta operators but defined slightly differently. Middleton and Goodwin devised a new operator called delta [Mi86], [Mi90], which shall be denoted as \( \delta_m \) throughout this chapter. This version of the delta operator is defined as

\[
\delta_m = \frac{z - 1}{T}.
\]

The work of Agarwal and Burrus [Ag75] was followed up by Goodall [Go89], [Go90], [Fo91] and led to a second definition of delta. This version of delta which shall be denoted as \( \delta_g \) is defined according to Agarwal and Burrus as

\[
\delta_g = z - 1.
\]

The question then arises: 'Which definition of the delta operator provides the best performance?' This chapter investigates the relative performances of the two definitions in an attempt to answer this question by looking at some of the properties of the operators and using error analyses. Equivalent digital filters implemented using the \( z \)-operator are used as a yardstick to compare the merits of the two delta operators.

5.2. Algorithmic error

Algorithmic error is the error introduced when a digital filter is derived, usually through emulation from its continuous equivalent [Ol95b]. It is defined as the difference between the output of the continuous filter and that of the emulated digital filter at each point of sampling. It is found that algorithmic error is dependent on the emulation technique used to derive the filter, and it decreases as the sampling rate is raised. When algorithmic error is determined, all finite wordlength effects are avoided.
and the error is independent of filter structure. Therefore algorithmic error will be the same for both forms of delta filter as well as the equivalent digital filter realised using the shift operator.

5.3. Coefficient representation error

Coefficient representation error is the discrepancy caused when the coefficients of an ideal digital filter are represented to finite accuracy. The error is evaluated by taking, at each sampling instant, the difference between the outputs of an ideal filter whose coefficients are represented with infinite precision and the equivalent filter with coefficients accommodated to fit a finite wordlength \( [F091] \), \([0195]a\). When evaluating coefficient representation error, the effects of multiplication roundoff are reduced to negligible proportions by simulating the behaviour of the filter using a high performance simulation package such as MATLAB.

Due to the definition of Middleton's delta operator, the coefficients of such a filter will be either equal to, or greater than those of Goodall, which in turn are either greater to or equal in size to those of a z-filter. When the coefficients are implemented with fixed-point arithmetic, where large numbers can be represented to a greater accuracy than small numbers, Middleton's definition will produce a better implementation than Goodall's, which in turn will produce a slightly better implementation than a filter that has been realised using the shift operator. Figure 5.1, which illustrates coefficient representation error for a fixed input signal of 1 radian per second, over a wide range of sampling values for a first order filter, and Figure 5.2 which shows equivalent results for a second order filter support these statements. In figures 5.1 and 5.2, the red plot indicates the peak coefficient representation error for the z-filter, blue for the delta filter realised using Goodall's definition of delta and green for the filter realised by Middleton's definition of delta. For all these results, the coefficients have been quantised to 12 fractional bits.

An interesting feature of Middleton's delta formulation is that, in addition to the
A comparison of alternative digital filter operators

explicit coefficients of the digital filter, there are also implicit coefficients, with the value \( T \), associated with each delta operation. This leads to improved accuracy as it has the effect of mimicking multiple-precision coefficients. A first-order filter realised with Middleton's version of delta and expressed in terms of that of Goodall produces

\[
H_m(\delta_m) = \frac{f_0 + f_1\delta_m^{-1}}{1 + g_1\delta_m^{-1}} = \frac{f_0 + f_1T\delta_g^{-1}}{1 + g_1T\delta_g^{-1}}
\]

(5.5)

and the equivalent filter implemented using Goodall's version of delta is

\[
H_R(\delta_g) = \frac{c_0 + c_1\delta_g^{-1}}{1 + r_1\delta_g^{-1}}
\]

(5.6)

Suppose the values \( f_0, f_1, g_1, T, c_0, c_1 \) and \( r_1 \) are represented by single precision words, then by comparison of equations (5.5) and (5.6) it is clear that \( f_0 \) and \( c_0 \) will be equal. The value \( c_1 \) in Goodall's version of the delta filter will be represented by a single precision word, whereas the equivalent value in Middleton's version is calculated from the product of two single precision words, \( f_1 \) and \( T \), to essentially mimic a word of double precision, which leads to greater accuracy in the coefficient representation. Clearly similar gains of accuracy are achieved when the \( r_1 \) coefficient in Goodall's definition is replaced by the product \( g_1T \) in Middleton's definition. For higher order filters the effect is more pronounced as multiple powers of \( T \) are used, thus mimicking triple or higher precision coefficients. (In practice however, this advantage is achieved at the expense of increased internal variable wordlength requirements.) On the whole for fixed-point representation, Middleton's version of the delta operator results in less error than that defined by Goodall, which in turn results in less coefficient representation error than that produced by the z-operator.

Figures 5.3 and 5.4 show graphs of coefficient representation error for first and second order filters respectively where the coefficients have been realised using floating-point arithmetic. For floating-point representations there is very little difference between the performance of the two delta realisations. Both delta filters produce significantly less coefficient representation error than the z-filter and clearly with floating point
A comparison of alternative digital filter operators

arithmetic the notion of using the z-operator can be seriously questioned. As the sampling interval, $T$, is varied, the coefficient representation error varies in a somewhat random manner depending on how accurately the coefficients are represented. This does, however lead to certain values of $T$ which Goodall’s version of delta yields less coefficient representation error than that of Middleton.

5.4. ADC quantisation error

In digital filtering applications a continuous signal needs to be sampled at each sampling instant by an ADC and converted to a finite wordlength which is representative of the value of the signal. The discrepancy between the two, after propagating through the digital filter constitutes the ADC quantisation error. This means that this error form is directly dependent upon the transfer function of the digital filter.

Figure 5.5 illustrates a general first-order z-filter with an input $u$ and output $y$, with coefficients specified to finite precision. Suppose the noise introduced by the quantisation in the analogue-to-digital conversion is $\mu$ then the corresponding error at the output is

$$ e_{\text{adc}}(z) = \mu(z)H_z(z) \quad (5.7) $$

where $H_z(z)$ is the transfer function of the first-order filter. Suppose the transfer functions of the equivalent delta filters realised using Goodall’s and Middleton’s definitions are respectively $H_g(\delta_g)$ and $H_m(\delta_m)$ then the ADC quantisation error is respectively given by

$$ e_{\text{adc} | \delta_g} = \mu(z)H_g(\delta_g) \quad (5.8) $$

and

$$ e_{\text{adc} | \delta_m} = \mu(z)H_m(\delta_m). \quad (5.9) $$

ADC quantisation error (and multiple-word truncation error) can be regarded as random.
A comparison of alternative digital filter operators

noise which there are several techniques for evaluating. The first technique is to
calculate a worst-case bound [Be58, Fr90] which can be excessively pessimistic. A
worst-case steady-state bound [SI64, Fr90] gives a more useful result, whilst the
determination of an r.m.s. value of the output error [Fr90] is the most reliable

technique.

As the three realisations of filter essentially implement the same function then the
ADC quantisation error will be approximately the same for all three filters; the only
difference will be caused by inaccuracies in the coefficient representation. Figure 5.6,
which shows the rms ADC quantisation error for an input quantisation level of 0.01
Volts, substantiates this remark. From the graph it can be seen that the ADC
quantisation error for the z-filter, denoted by (o), and the errors for Goodall's delta (x)
and Middleton's delta (+) are approximately equal. Also it can be observed that as the
sampling rate is increased the ADC error decreases. From the work on stochastic noise
presented in Chapter 4, as the sampling rate is raised, the size of the residues of the
digital transfer function decrease, thus leading to a decrease in the magnitude of the
error.

5.5 Multiple-word truncation error

Multiple-word truncation error arises following multiplications in the internal
computations of recursive digital filters, as multiple-wordlength products will be
truncated or rounded to shorter values. This results in quantisation noise which, after
propagating through the digital filter, constitutes the multiple-word truncation error.

Consider a general first-order z-filter as illustrated in Figure 5.7. There is just one
source of multiple-word truncation within the internal computations, that is in the
calculation of the variable \( v_0 \). (All circumflexed variables denote the quantised
equivalents.) The quantisation noise \( e_v \), generated at each sampling interval will
propagate through to the output of the digital filter. (The quantisation of the output
through digital-to-analogue conversion is not considered in the following analyses as
A comparison of alternative digital filter operators

it will be of the same order of magnitude for all three filter structures.) As the system of Figure 5.7 is linear the noise source $e_v$ may be modelled as an input to the system and by inspection, the transfer function relating the noise source with the output is the transfer function of the filter. Therefore

$$e_{m|u}(z) = e_v(z)H_s(z)$$

(5.10)

where $H_s(z)$ is the transfer function of the digital filter.

Consider a digital filter implemented using the delta operator of Goodall; the canonical form given in Figure 5.8. Again in the internal computations there is just one source of quantisation error which occurs in the computation of $v$. Using the principles of the previous analysis it follows that the error $e_{m|0}$ is given by

$$e_{m|0}(z) = e_v(z)H_s(z)$$

(5.11)

Consider finally a digital filter implemented using Middleton's delta operator. There are two sources of quantisation which affect the internal computations. The first, as in the other two structures, lies in the computation of the internal variable $v$. The second lies in the computation of the internal variable $w$ as a consequence of the delta operator which is

$$w(k+1) = w(k) + T_q v(k)$$

(5.12)

where $T_q$ is the value of the sampling interval, stored to a finite number of bits, and $w(k)$ denotes the $k^{th}$ sample of $w$.

The computation of the error due to quantisation of $v$, $e_v$, can be calculated in a similar manner to that of the $z$-filter or Goodall’s delta filter. That is

$$e_v(z) = e_v(z)H_m(z)$$

(5.13)

where $H_m(z)$ is the transfer function of the digital filter realised by Middleton's delta.

Figure 5.9 shows how the component of the output error $e_w$ is related to the
quantisation from the delta operation, \( e_n \). As the system is linear, this can be analyzed in absence of the input signal and the other noise source. By inspection of the simplified diagram of Figure 5.9

\[
e_n(z) = (v(z)F_0 + \hat{w}(z)F_1)
= \hat{w}(z) (f_1 - F_0 g_1).
\]  
(5.14)

Since

\[
\hat{w}(z) = e_n(z) \delta_g^{-1} + v(z)T_q \delta_g^{-1}
= e_n(z) \delta_g^{-1} - \hat{w}(z)g_1 T_q \delta_g^{-1}
= e_n(z) \delta_g^{-1} - \hat{w}(z)g_1 \delta_m^{-1}
\]
(5.15)

the component of the overall error due to the quantisation incurred during the delta operation is

\[
e_n(z) = e_n(z) \left\{ \frac{1}{T} \cdot \frac{(f_1 - F_0 g_1) \delta_m^{-1}}{1 + g_1 \delta_m^{-1}} \right\}.
\]  
(5.16)

The total multiple-word truncation error generated by Middleton's version of the delta filter is therefore the sum of the two components, given in equations (5.13) and (5.16). That is

\[
e_{m\\\|n}(z) = e_n(z) + e_n(z)
= e_n(z) \left\{ \frac{1}{T} \cdot \frac{(f_1 - F_0 g_1) \delta_m^{-1}}{1 + g_1 \delta_m^{-1}} \right\}
\]  
(5.17)

By inspection of equations (5.10) and (5.11) it is clear that the z-filter and the equivalent digital filter realised using Goodall’s definition of delta will generate similar amounts of multiple-word truncation error; the only difference being in terms of the representation of the filter coefficients. The first term in the expression for multiple
word truncation error in Middleton's delta filter (equation (5.17)) is approximately equal to that generated by an equivalent filter realised either in z or Goodall's delta. However as a result of quantisation generated in Middleton's delta operation, there is an extra term which contributes to the overall error, thus degrading the performance.

For internal variables employing 8 underflow bits, Figure 5.10 shows how the multiple-word truncation error varies as the sampling frequency is raised. Again, the results of the z-filter are denoted by 'o', the results from Goodall's delta filter, 'x' and '+' denotes the error from Middleton's delta filter. For z- and δ-filters the multiple-word truncation error decreases as the sampling rate is raised as a consequence of diminishing filter residues. It is also clear that the z- and δ-filters out-perform the delta filter of Middleton, whose performance deteriorates for faster sampling rates. Figure 5.11 illustrates the two constituent components of multiple-word truncation error as generated by Middleton's equivalent delta filter. The magenta line represents the error generated by the first term of equation (5.17) which, as predicted, is similar to the total error generated by the z- and δ-filters. The black line indicates the error generated by the second term of the expression. As the sampling frequency is raised this error increases, thus degrading the overall error response.

From these results, it is clear that Middleton's definition gives inferior multiple-word truncation error than the other two forms and therefore leads to increased internal variable requirements.

5.6. Coefficients

From equation (5.3) which approximates digital differentiation, as T becomes smaller it is clear that Middleton's delta operator approximates to the Laplace operator, s. The coefficients of a digital filter when implemented using this delta operator will approximate to the coefficients of the equivalent continuous filter. This effect does not occur with either the shift operator or with Goodall's definition of delta.
5.7. Speed of computation

For a first order filter realised in canonical form, each computation requires three multiplications and two additions if the shift operator is implemented. Goodall's definition of delta requires three multiplications and three additions, whilst Middleton's realisation takes three additions but four multiplications. As multiplication is computationally the most time consuming arithmetic operation that takes place when implementing digital filters, a digital filter realised using the shift operator will clearly be the fastest realisation. Goodall's definition requires just one extra addition in comparison to the z-filter and therefore the computation speed will not be significantly slower than that of the shift operator. Middleton's delta filter, with the extra multiplication will clearly be the slowest implementation which could be unsuitable for applications geared to fast sampling.

5.8. The Modified Canonic Filter Architecture

A modification can be made to the filter architecture of Goodall's canonic delta filter by moving the feedback coefficients into the forward path of the filter and scaling the remaining coefficients appropriately. The structure of the general modified canonic second-order transfer function of

\[ H(\delta_s) = \frac{p + qd_1\delta_s^{-1} + r d_2 \delta_s^{-2}}{1 + d_1 \delta_s^{-1} + d_2 \delta_s^{-2}} \]  

(5.18)

is illustrated in Figure 5.12. One of the major advantages of the modified canonic structure is that the maximum magnitudes of the internal variables will be of similar order to that of the input.

Figure 5.15 shows how the coefficient representation error varies for a second-order filter realised using Goodall's canonic and modified canonic delta structures. For this case the coefficients have been specified to twelve fractional bits. It can be seen that the use of the modified canonic structure leads to a reduction of the error. The principle which explains why Middleton's canonic delta structure generates less
A comparison of alternative digital filter operators

coefficient representation error than Goodall's equivalent, can also be applied to this situation.

Unlike a standard canonic delta or z-filter, no multiple-word truncation error is generated in the computation of the internal variable v. However as multiplication is used in the computation of the internal variables w and x then these become the sources of multiple-word truncation error.

Consider a first-order modified canonic delta filter. The internal variable w is computed from the multiplication of the internal variable with coefficient $d_1$. The result of the multiplication will be reduced to a lesser length. Suppose the noise introduced in the computation of w is denoted $\varepsilon_w$ as in Figure 5.13 then it can be shown that the multiple-word truncation error is given by

$$e_{m/w}(z) = e_w(z) \left\{ \frac{(q-p)\delta_x^{-1}}{1 + d_1\delta_x^{-1}} \right\}$$  \hspace{1cm} (5.19)

An interesting adjustment to the modified canonic architecture can be made by interchanging the coefficient $d_1$ and the delta operator in a first-order filter. ($d_2$ and the delta operator should also be exchanged for the case of a second-order filter.) In such an arrangement the internal variable w is calculated using

$$w'(z) = w'(z)z^{-1} + x(z)z^{-1}$$  \hspace{1cm} (5.20)

and

$$w(z) = d_1w'(z)$$  \hspace{1cm} (5.21)

as shown in Figure 5.14. It can be shown for a discrepancy $e_w$ in the computation that the multiple-word truncation error is given by

$$e_{m/w_2}(z) = e_w(z) \left\{ \frac{q-p}{1 + d_1\delta_x^{-1}} \right\}$$  \hspace{1cm} (5.22)
A comparison of alternative digital filter operators

which differs from equation (5.19) only by the absence of the $\delta_e^{-1}$ term in the numerator.

Suppose the original modified canonic form is denoted Class I and the altered form denoted Class II then Figure 5.16 shows how the multiple-word truncation error varies with sampling rate for both implementations of the modified-canonic architecture. In the Class I implementation the error increases as the sampling rate is raised. In the Class II architecture the error is lower than that of the Class I structure but remains approximately constant as the sampling rate is altered. These trends can be explained by referring to equations (5.19) and (5.22). The noise in the Class I architecture passes into the $\delta_e^{-1}$ block which acts as an accumulator to effectively amplify the noise. For faster sampling rates more samples of noise are accumulated per unit time leading to increased error. However, the noise source in the Class II architecture is generated after the delta operator so the accumulation effects do not occur.

Although both forms of modified canonic filter generate less coefficient representation error than Goodall’s standard canonic filter, it is at the expense of increased multiple-word truncation error (or increased internal variable wordlength requirements).

5.9. Summary

This chapter has presented some advantages and disadvantages of implementing a digital filter using a delta operator against the well-used shift operator. Algorithmic error and ADC quantisation do not play a significant role when the choice of operator is to be determined. For fixed-point arithmetic, Middleton’s definition provides, in general, lower values of coefficient representation error than an equivalent $z$-filter or a filter realised using Goodall’s definition of delta. However this advantage is offset by the additional noise introduced by multiple-word truncation error as a consequence of the additional multiplication. For floating-point arithmetic it has been shown that both implementations of delta produce significantly less coefficient representation error than their shift counterparts; Middleton’s definition being only slightly superior to that
of Goodall. The beauty of Middleton's version of delta is that for fast sampling rates, the coefficients of the corresponding digital filter approximate their continuous counterparts. This is a useful facility for the control designer who, in general, will be more familiar with working in the s-plane than in the discrete domain.

If a filter requires realisation using fixed-point arithmetic and relatively slow sampling rates then Middleton's delta operator will be the most suitable operator to use. In situations where fast sampling is required, Goodall's definition of delta should be implemented, especially if floating-point arithmetic is available.

In order to further reduce the coefficient representation error using Goodall's definition of delta, the modified canonic structure should be used. The original modified canonic architecture reduces the coefficient representation error but the multiple-word truncation error becomes more problematic especially for increasing sampling rates. However interchanging the coefficient with the delta operator in the internal variable computations minimises this problem.
Figure 5.1. Coefficient representation error for fixed-point, first-order filters.

Figure 5.2. Coefficient representation error for fixed-point, second-order filters.
Figure 5.3. Coefficient representation error for floating-point, first-order filters.

Figure 5.4. Coefficient representation error for floating-point, second-order filters.
A comparison of alternative digital filter operators

Figure 5.5. Block diagram illustrating the generation of ADC quantisation error.

Figure 5.6. Variation of ADC error with sampling rate for three types of filter operator.
A comparison of alternative digital filter operators

Figure 5.7. The source of multiple-word truncation error in a z-filter.

Figure 5.8. The source of multiple-word truncation error in Goodall's δ-filter.

Figure 5.9. The second source of multiple-word truncation error in Middleton's delta filter.
Figure 5.10. Total average multiple-word truncation error for all three filter operators.

Figure 5.11. Average multiple-word truncation error components for Middleton's delta operator.
A comparison of alternative digital filter operators

Figure 5.12. The original structure for Goodall's modified-canonic delta filter.

\[ w(z) = w(z)z^{-1} + d_1v(z)z^{-1} \]

Figure 5.13. Updating the internal variable in the original Class I modified-canonic delta filter.

\[ w'(z) = w(z)z^{-1} + v(z)z^{-1} \quad w(z) = d_1w'(z) \]

Figure 5.14. Updating the internal variable in the Class II modified-canonic delta filter.
A comparison of alternative digital filter operators

Figure 5.15. Coefficient representation error for a second-order filter realised using Goodall's standard and modified-canonic architectures.

Figure 5.16. Multiple-word truncation error for a first-order filter realised using the Class I and Class II modified canonic architectures.
CHAPTER SIX
HARDWARE DESIGN

6.1 Introduction

This chapter describes the hardware that was constructed to verify the theoretical aspects of the research in practice.

The first aspect that will be considered is the design of digital filter hardware that can be interfaced to an IBM compatible personal computer. The main feature of this filter hardware is that the sampling interval can be varied by setting the time on an array of switches.

Several extra circuits will be considered that can emulate the behaviour of certain first- and second-order physical systems. These will be used later in the thesis as plant models to assess the performance of digital control algorithms under varying conditions of sampling rate along with coefficient and variable wordlength.

Full circuit diagrams for the hardware are shown in Appendix C.
6.2. Digital controller design

6.2.1. PC Interface

The digital controller hardware is connected to an IBM compatible personal computer using a commercially available input / output card. The card, which is based upon the Intel 8255 parallel interface adaptor (PIA), is available in kit form from "Maplin MPS". The card provides the facility for three 8-bit ports, Port A, Port B and Port C, which can be configured as either input or output ports.

For this project, the card is programmed such that port A is an input port (for connecting an analogue-to-digital converter) and port B is an output port (for connecting a digital-to-analogue converter). Port C is configured such that 4 bits are used for input and the remaining bits are used for output. This port is used for general handshaking.

6.2.2. Timer

A facility for adjusting the sampling interval of the digital filter is included in the circuit design. The value of the sampling interval can be adjusted by setting the time on an array of twelve binary switches; the lowest significant bit corresponds to 0.0001 seconds (10 kHz).

Figure 6.1 illustrates the general block diagram of the timing circuitry. An 8 MHz crystal oscillator is used to generate the oscillations. However, the signal produced by the oscillator consists of very small triangular-shaped oscillations superimposed upon a d.c. signal [Pe89]. Obviously such a signal would not vary sufficiently to trigger TTL logic. A simple transistor amplifier is used to remove the dc component, and amplify the fluctuations to a level where TTL logic can be triggered.

A cascade of frequency dividers is used to reduce the 8 MHz signal frequency to 10 kHz. A TTL divide-by-8 frequency divider uses the 8 MHz triangular oscillations and generates a 1 MHz square wave. Two cascaded divide-by-10 frequency dividers reduce
the frequency by a factor of one hundred to 10 kHz.

The 10 kHz signal is then passed to a 12-bit binary counter that counts the number of 0.0001 second pulses received. The 12-bit word held on the counter is denoted as Word A. Word B is the preset value, set on the binary switches, to represent the required sampling interval. A 12-bit comparator, composed of discrete logic, is used to compare the magnitudes of Words A and B and when both words are equal then the required sampling time has been achieved. A pulse is then sent to trigger an analogue-to-digital conversion and simultaneously reset the binary counter.

A monostable multivibrator is used to 'stretch' the pulse so that its duration is sufficient to trigger the analogue-to-digital converter.

6.2.3. Digital Controller Hardware

The digital control hardware essentially consists of three main elements: a digital-to-analogue converter, an analogue-to-digital converter and some control logic. These constituents have been designed to interface directly to the input / output PC card. The general block diagram of this scheme is illustrated in Figure 6.2.

The digital-to-analogue converter is based around the Ferranti ZN426 8-bit DAC. The digital input to the digital-to-analogue converter is provided by Port B. The output voltage from the digital-to-analogue converter is subjected to a resistor and operational amplifier network to adjust the gain and offset as laid out in the Ferranti data sheet [Fer82]. The gain and offset were adjusted such that the output resolution of the digital-to-analogue converter is 40 millivolts and the possible output range is symmetrically bipolar. This enables the output range to vary from just under -5 Volts (for a binary input of 00000000) to just over +5 Volts (for a binary input of 11111111).

The analogue-to-digital converter is based around the Ferranti ZN448 8-bit successive
approximation ADC that has an in-built clock oscillator. The digital output from the ZN448 is connected to Port A. The analogue input voltage is first buffered before being adjusted using the gain and offset adjustment as documented in the Ferranti data sheets [Fer80a, Fer80b]. The gain and offset were adjusted such that the converter has a resolution of 40 millivolts and is therefore complementary to the digital-to-analogue converter. Both the digital-to-analogue converter and the analogue-to-digital converter have built-in precision voltage references. However, the voltage reference input for the analogue-to-digital converter is connected to the reference output for the digital-to-analogue converter for two reasons: First this reduces the component count and secondly, both converters have the same input voltage reference to help match the converters.

The analogue-to-digital conversion is triggered by a pulse from the digital timer. When the conversion is complete an "End Of Conversion" signal is generated by the analogue-to-digital converter. Some control logic, implemented by a D-type flip-flop is used to monitor this signal. Initially a signal from the output section of Port C, generated from the digital controller hardware, resets the flip-flop. When the end-of-conversion signal is generated, it triggers and sets the D-type flip-flop. The output of the flip-flop is sent to the input portion of Port C. This signal is continuously polled by the control software to detect when the sampling has been completed.

6.2.4. Digital Controller Software

Figure 6.3 shows the flow diagram of the software, written in Turbo C, to operate the digital control system. This diagram places very little emphasis on the control algorithms but concentrates mainly on the interaction between the personal computer and the hardware.

When the software is executed, the first stage is to initialise all the ports. This establishes the port addresses for the three ports on the input / output card, and configured Port A as an input, Port B as an output and Port C as a combined input /
output port. The next stage is to send the value 128 to Port B. This means that the output from the digital-to-analogue conversion network is zero Volts ensuring that no spurious control effort takes place.

The user is prompted to enter the digital filter coefficients, the value of T and the total required duration of filtering. The user will also be able to trigger the start of the filtering.

Following this, a pulse is generated and sent to Port C to reset the control logic. The output from the control logic (which is now zero) is fed to the input nibble of Port C.

By counting the number of samples that have taken place the program will determine whether the required duration for filtering has elapsed. If it has then the program will terminate.

After an analogue-to-digital conversion has taken place, i.e. at every sampling instant, the state of the input of Port C will change from zero to one. The input nibble from Port C is continuously polled until this change has occurred.

Once this change has been detected, the filter algorithm computes the output value to be sent to the digital-to-analogue converter and the filter variables are then updated. It is necessary to send the output word to the digital-to-analogue converter before updating the filter operators to minimise the delay between the start of the sample and the output being generated. Once this has been completed, the control logic is reset and the cycle repeats.
6.3. Plant Models

This section describes the design of various circuits to emulate the behaviour of low-order plant models. These circuits were constructed using nearest-value components and no attempt was made to fine-tune the circuits to their exact mathematical model. The uncertainties in the exact parameters in the plant model make the situation realistic because the characteristics of real physical systems tend to drift with time and temperature.

6.3.1. Model Of A Closed-Loop Control System

Figure 6.4 illustrates a block diagram for two closed-loop control systems. The top portion of the diagram shows the block diagram for an analogue control system. The lower section of the diagram shows the block diagram for a digitally controlled equivalent. This provides a crude method to practically compare the performance of analogue and digitally controlled systems.

This system can readily be modelled with networks of operational amplifiers, resistors and capacitors. The error amplifiers can be modelled using operational amplifiers as inverting summing junctions.

Two identical plant models were chosen to be first-order lags with transfer function

\[ P(s) = \frac{1}{s + 1}. \]  

(6.1)

The first section of Figure 6.7 represents a first-order lag. From this circuit diagram it can be deduced that the transfer function of the network is approximately

\[ P(s) = \frac{1}{sCR + 1}. \]  

(6.2)

where C and R are respectively the capacitor and resistor values. C was chosen to be 6.8 \( \mu \)F and R was chosen to be 150 k\( \Omega \). This gives a time constant of nearly one second. The capacitors were unpolarised metallised polyester, specified as having 10%
tolerance. Fortunately the values of the capacitors for both plants were within 1% of each other such that the parameters of both plant models would be similar.

The compensator for the analogue section was chosen to be a proportional plus integral (PI) controller. The transfer function of the PI circuit given by Figure 6.5 is approximately

\[ C(s) = \frac{sC_2R_2 + 1}{sC_2R_1} \]  

(6.3)

The transfer function of the PI compensator, which gives an adequate phase margin of 58.7 degrees at a cross-over frequency of 4.9 radians per second, is

\[ C(s) = \frac{0.22s + 1}{0.06s} \]  

(6.4)

leads to components with nearest values of 27 kΩ for R₁, 100 kΩ for R₂ and 2.2 μF for C₂.

6.3.2. Second-Order Lightly-Damped System

Figure 6.6 shows the circuit diagram [Wi88] for a second-order system with an approximate transfer function of

\[ P(s) = \frac{1}{C_1C_2R_1R_2s^2 + C_2R_2s + 1} \]  

(6.5)

A system with transfer function

\[ P(s) = \frac{10}{s^2 + s + 10} = \frac{1}{0.1s^2 + 0.1s + 1} \]  

(6.6)

gives responses that exhibit lightly-damped oscillations. To realise this transfer function C₂ was chosen to be 1 μF and R₂ was selected to be 100 kΩ such that the product C₂R₂ is equal to 0.1. R₁ was chosen to be 147 kΩ (a 100 kΩ and a 47 kΩ resistor in series) and C₁ was chosen to be 6.8 μF such that the product C₁R₁ is approximately equal to one.
6.3.3. Second-Order Type I System

A second-order Type I system, consisting of a simple lag cascaded with an integrator, is illustrated in Figure 6.7. The approximate transfer function of the system is

\[ P(s) = \frac{1}{sC_1R_1 + 1} \cdot \frac{1}{sC_2R_2}. \]  \hspace{1cm} (6.7)

A second-order plant whose transfer function is

\[ P(s) = \frac{10}{s(s + 1)} \]  \hspace{1cm} (6.8)

can be approximated by setting \( C_1 \) to 6.8 µF, \( R_1 \) to 150 kΩ, \( C_2 \) to 1 µF and \( R_2 \) to 100 kΩ. In order to reset the integrator, a switch was placed across \( C_2 \) to discharge it.

6.4. Testing

This section provides evidence to show that the components of the hardware function correctly.

To verify that the digital control hardware functions correctly a sinusoidal signal was applied to the input of the digital controller. This signal was sampled at every sampling instant and directly sent to the output of the controller. Figure 6.8, which shows the oscilloscope displays of the input sinusoid and the output of the controller, suggests that the digital controller functions correctly.

To verify that the closed-loop control model of Figure 6.4 functions correctly, a unit step was applied to the system. The digital controller was replaced by a wire link so that the lower portion of the system resembled the uncompensated closed-loop system. Figure 6.9(a) illustrates the theoretical simulated step responses of the compensated and uncompensated systems. Figure 6.9(b) shows the oscilloscope display when the system was subject to a one Volt step input. The practical responses agree closely with the theoretical responses.
The correct functioning of the remaining systems was verified by applying a one Volt step to the respective system inputs. Figures 10(b) and 11(b) show the oscilloscope printouts for the responses of the lightly damped system and the Type I system. Figures 10(a) and 11(a) show the theoretical simulated unit step responses for the respective systems, verifying that the practical systems function as expected.

6.5. Summary

This chapter has outlined the design of various items of hardware that will be applied later to verify the practical results of the research. The full circuit diagrams are shown in Appendix C.

A digital filter has been designed where the sampling rate can be adjusted by setting the required sampling interval on an array of switches. If such an arrangement was to be commercially produced, the contents of the timer would likely to be accommodated on a single integrated circuit (such as a programmable logic device). In addition, if the system was to be commercially produced, the sampling time could be selected using software via the output of a microprocessor, microcontroller, DSP or computer.

Simple operational amplifier circuits have been constructed to simulate the behaviour of various first- and second-order systems. Their behaviour corresponds closely to the theoretical responses; any discrepancy is caused by non-ideal component values and effects such as stray resistance, capacitance and inductance. However, this is practically acceptable as there are always uncertainties in determining the exact transfer function of a physical system.

Figures 6.12 and 6.13 respectively show digital photographs of some of the circuitry and the entire experimental set-up.
Figure 6.1. Block diagram of timer to set the sampling interval of the digital controller

Figure 6.2. Block diagram showing the overall scheme of the digital control hardware.
Figure 6.3. Flow chart illustrating the operation of the digital filter.
Figure 6.4. Block diagram illustrating how the outputs of an analogue and a digital control system may be compared.

Figure 6.5. Basic circuit diagram of a P+I controller

Figure 6.6. Basic circuit diagram of a second-order lightly-damped system

Figure 6.7. Basic circuit diagram of a Type I system
Figure 6.8. Continuous sinewave and output of unity-gain digital filter.

Figure 6.9(a). Theoretical responses

Figure 6.9(b). Practical responses

Figure 6.9. Responses of compensated and uncompensated systems to a one Volt step input
Hardware Design

Figure 6.10(a). Theoretical response

Figure 6.10(b). Practical response

Figure 6.10. Response of a lightly damped second-order system to a 1 Volt step input.

Figure 6.11(a). Theoretical response

Figure 6.11(b). Practical response

Figure 6.11. Response of a Type I, second-order system to a one Volt step-input
Figure 6.12. Digital photograph of the digital timer, the digital controller hardware and a plant model.

Figure 6.13. Digital photograph of the practical set up.
CHAPTER SEVEN
DIGITAL FILTER ERROR ANALYSIS USING D.F.T. TECHNIQUES

7.1 Introduction

Many methods have been devised to analyze the various error components that are generated by digital filters. The error forms can be grouped into two categories: deterministic error, $e_d$, and stochastic error, $e_s$.

Deterministic error itself consists of two components: algorithmic error, $e_a$, which is the error introduced in the transition from the s-domain to the z-domain [Ol95b], assuming the filter is derived through emulation, and coefficient representation error, $e_c$, the error introduced when the filter coefficients are specified to finite accuracy [Ol95a, Fo89, Fo91]. These two error mechanisms can be combined to give a quantity known as 'quantised-coefficient algorithmic error' [Ol95a] which for a given input signal can readily be computed.

Stochastic error also contains two components, both finite wordlength effects. The first $e_q$ is due to ADC quantisation error [Fo89, Kn94]. The ADC samples a continuous waveform and converts it to a digital value of fixed wordlength. The difference between the continuous and the quantised digital values, after propagating through the
discrete controller, constitutes $e_q$. The second component, multiple-word truncation error $e_m$, is a discrepancy which occurs when the result of a multiple-word arithmetic operation, such as multiplication, is stored in shortened form [Fo89, Kn94]. Both $e_q$ and $e_m$ can be modelled as random variables and hence stochastic analysis may be undertaken on both of these error effects.

Some texts [e.g. Kn94] model $e_q$ as a stochastic error in so far as the values of the coefficients vary in a somewhat random manner with respect to changes in the value of the sampling interval. However the work presented in this chapter will model it as a deterministic quantity.

Each of the four error components $e_s$, $e_o$, $e_q$ and $e_m$ can be investigated in isolation using simulation techniques though in practice all the error forms will coexist. This chapter presents a technique which uses spectral analysis to separate the total deterministic error from the stochastic error. It is assumed that the multiple-word truncation error $e_m$ has been reduced to a negligible value, which is a reasonable assumption if attention is given to the filter structure and internal wordlength requirements [Fo91]. This is also a realistic situation if the digital filter is implemented using a high-level language with high-precision floating point arithmetic.

This chapter commences by investigating techniques that are widely used to determine the frequency content of time-varying signals. Fourier analysis may be used for continuous periodic signals, whilst the Fourier Transform is used to analyze aperiodic continuous signals. The Discrete Fourier Transform is also presented to determine the frequency spectrum of discrete-time signals.

A slightly modified D.F.T. technique is introduced in this chapter which can be used to determine and reconstruct the sinewave given by a single spectral component. This modified D.F.T. technique may be used in digital filter error analysis to isolate the deterministic and stochastic error components and reconstruct the individual components. Several examples are given which demonstrate the effectiveness of this
method. However, a degree of care needs to be taken in order for results of high accuracy to be obtained.

7.2 Spectral Analysis

In signal processing applications, there are several specific techniques that may be used to analyze the spectral content of time-varying signals.

For continuous signals that are considered periodic over all values of time, the Fourier Series may be used to determine the individual spectral contents of that waveform [St91, Co82]. The Fourier Series of a continuous periodic waveform consists of an infinite number of frequency components which occur at intervals of \( \omega \) radians per second. Therefore the Fourier Series leads to a discrete spectrum.

The Fourier Series analysis is only applicable to periodic signals. For non-periodic waveforms, such as a single pulse, the Fourier Transform needs to be used [Co82, Me91]. For single waveforms, the period can be considered to be infinitely long. The expression for the Fourier Transform can be derived from the Fourier Series by making the assumption that \( u(t) \) is initially periodic.

The shift theorem is a useful tool for Fourier transform analysis. This states that the Fourier transform \( U_2(\omega) \) of a signal \( u(t) \) which has been delayed by \( \tau_d \) seconds is equal to the Fourier transform \( U_1(\omega) \) of the undelayed signal \( u(t) \) delayed by a phase factor [Co82].

\[
U_2(\omega) = U_1(\omega)e^{-j\omega\tau_d}
\]  

(7.1)

The Fourier Transform can be applied to non-repetitive continuous signals. However, for signals which have been sampled at discrete intervals of \( T \) seconds, the Fourier Transform can be modified to give the Discrete Fourier Transform (DFT) [Co82, Te84]. As the function \( u(t) \) is a non-repetitive waveform, its duration is therefore finite
and can be sampled over, say, N T-second sampling intervals. The discrete samples $u(nT)$ can be expressed as a sequence of impulse functions, namely

$$u_d(t) = \sum_{n=0}^{N-1} u(n) \delta(t-nT). \quad (7.2)$$

Despite $u_d(t)$ containing an infinite number of frequency components, it is impractical to evaluate the frequency spectrum for each and every frequency value. A finite set of N frequencies is usually considered where each spectral component is separated by $\Omega$ radians per second from its neighbouring component. Assuming $\omega_s$ is the angular sampling frequency, $\Omega$ may be determined using

$$N\Omega = \omega_s = \frac{2\pi}{T}. \quad (7.3)$$

Therefore

$$\Omega = \frac{2\pi}{NT}. \quad (7.4)$$

The Fourier Transform $U(\omega)$ of $u_d(t)$ is

$$U(\omega) = \int_{-\infty}^{\infty} u_d(t) e^{j\omega t} dt = \int_{-\infty}^{\infty} \sum_{n=0}^{N-1} u(nT) \delta(t-nT) e^{j\omega t} dt. \quad (7.5)$$

If $\omega = m\Omega$ where $m = 0, 1, ..., N-1$ then

$$u(m\Omega) = \sum_{n=0}^{N-1} u(nT) \int_{-\infty}^{\infty} \delta(t-nT) e^{j\omega t} dt. \quad (7.6)$$

The integrand of equation (7.6) represents the Fourier Transform of an impulse delayed by $nT$ seconds, i.e. $e^{j\omega t}$. So the discrete Fourier transform of N discrete samples, $u(nT)$, is
Digital Filter Error Analysis Using DFT Techniques

\[ U(m\omega) = \sum_{n=0}^{n+N-1} u(nT) e^{-jn\omega nT} \tag{7.7} \]

The inverse Discrete Fourier Transform is given by

\[ u(nT) = \frac{1}{N} \sum_{m=0}^{m+N-1} U(m\omega) e^{jn\omega nT} \tag{7.8} \]

### 7.3 A Normalised DFT Algorithm

There are many simulation packages which can be used to determine the Discrete Fourier Transform of discretely sampled signals. This is usually carried out using the Fast Fourier Transform (FFT). For example, MATLAB [Ma92] has a built in FFT function which determines the Fast Fourier Transform of a vector of samples, along with an inverse FFT algorithm. The problem with such algorithms is that they operate over discrete frequency values which may or may not contain a specific frequency of interest. It is therefore necessary to devise a slightly modified DFT and inverse DFT algorithm to cope with such a situation.

Consider the z-transform of a finite sequence of N samples, taken from a signal \( u(t) \):

\[ U(z) = \sum_{k=0}^{k+N-1} u(kT) z^{-k} \tag{7.9} \]

where \( T \) is the duration of the sampling interval. To analyze the spectral content, by examining a sinusoid of angular frequency \( \omega \), which is a constituent of \( u(kT) \), the substitution \( e^{j\omega T} \) is made for \( z \) in equation (7.9). In this work a 'normalised' discrete Fourier transform, NDFT, was achieved by dividing by the number of samples and multiplying by \( 2j \) to give

\[ U(\omega, T) = \frac{2j}{N} \sum_{k=0}^{k+N-1} u(kT)e^{-j\omega k} \tag{7.10} \]

This method of determining the NDFT necessitates that:
ensuring that an integer multiple of half cycles is used in the NDFT computation.

To reconstruct a sinewave of angular frequency \( \omega \) from its NDFT, \( U(\omega, T) \), the following inverse NDFT algorithm is used to create an N-point discrete-time sequence:

\[
U(kT) = \sum_{n=-N/2}^{N/2-1} \left\{ \frac{U_n e^{j n \omega T} + \overline{U}_n e^{-j n \omega T}}{2j} \right\} z^{-k}
\]

For example a sinewave of the general form

\[
u(t) = Msin(\omega t + \phi)
\]

has an NDFT

\[
U(\omega, T) = Mz^\phi
\]

using this technique. The factor 2j in equation (7.10) ensures that the amplitude and phase in equation (7.14) are identical to those in equation (7.13).

### 7.3.1 Example: Computing the NDFT

To illustrate the process the sinewave \( u(t)=Msin \omega t \) is considered for half a cycle, where \( M=1.2 \) and \( \omega=\pi \) rad/sec. Suppose a total of \( N=4 \) samples of this waveform is taken where the sampling interval \( T=0.25 \) seconds; the criterion of Equation (7.11) is then satisfied as these four samples constitute exactly one half cycle. Table 7.1 illustrates the computation.

The same result is found for any value of N that meets the criterion of equation (7.11), for this waveform \( u(t) \).
Digital Filter Error Analysis Using DFT Techniques

<table>
<thead>
<tr>
<th>k</th>
<th>t=kT (secs)</th>
<th>u(kT)</th>
<th>$e^{-j\omega kT}$</th>
<th>$u(kT) e^{-j\omega kT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$1 - j0$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.8485</td>
<td>$0.7071 - j0.7071$</td>
<td>0.6 - j0.6</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>1.2</td>
<td>$0 - j1$</td>
<td>0 - j1.2</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>0.8485</td>
<td>$-0.7071 - j0.7071$</td>
<td>-0.6 - j0.6</td>
</tr>
</tbody>
</table>

$$U(\omega, T) = \frac{2J}{4} \sum_{k=0}^{k=3} u(kT)e^{-j\omega 2\pi kT} = 2j(0 - 0.6j) = 1.2 = M \angle 0$$

Table 7.1. Computing the NDFT of $u(t)=1.2 \sin \pi t$

7.4. Definitions

Consider the system of Figure 7.1 which illustrates how the total error is generated by a digital filter. An analogue filter $F(s)$ is the datum upon which the ideal filter is based. It is presumed that the reference filter is continuous but it could be an ideal digital filter, $G(z)$. Suppose the input to the filter is $u(t)$ then the corresponding output, measured every $T$ seconds is $y_1(kT)$. The output from the reference filter should be computed using a simulation package such as MATLAB to produce accurate datum results and to eliminate all possible effects of drift associated with a practical continuous filter. As this technique generates $u(kT)$ and $y_1(kT)$ purely by simulation, it may be performed as an 'off-line' process.

The input $u(t)$ is supplied to a practical digital filter operating in real-time with the input sampled every $T$ seconds and subject to quantisation by an analogue-to-digital converter to give a filter input $u_q(kT)$. The digital filter $H(z)$, with coefficients specified to finite accuracy, will generate an output value $y_3(kT)$ assuming $e_m$ is negligible. The values of $u_q(kT)$ and $y_3(kT)$ can be stored as vectors in the memory of the hardware for processing later.
Digital Filter Error Analysis Using DFT Techniques

The difference between the ideal filter output \( y_1(kT) \) and the practical digital filter output \( y_3(kT) \) gives the total error value \( e_{\text{tot}}(kT) \) at each sampling instant, \( T \).

Figure 7.2 shows a more detailed model which illustrates how the stochastic and deterministic error forms are individually generated. The digital filter \( H(z) \) is considered in two ways: with its input sampled but not quantised, and quantised to a finite number of bits. Thus \( y_3(kT) \) and \( y_3(kT) \) are generated. Together with \( y_1(kT) \) they can be combined in various ways, as shown in the figure, to model the generation of deterministic and stochastic errors.

### 7.5. Analysis

#### 7.5.1. Total Error Definition

From the four sets of number sequences \( u(kT), y_1(kT), u_q(kT) \) and \( y_3(kT) \) (see Figure 7.2) it is possible, using spectral analysis, to evaluate the deterministic and stochastic errors. It is important, however, in all the computations that follow to use only the parts of the output sequences that are in the 'residual state' (a.c. steady state) otherwise any remaining transient effects will invalidate the results.

The first stage is to take \( N \) points from the theoretical residual sequence \( y_1(kT) \) and the corresponding theoretical input sequence \( u(kT) \) where the value of \( N \) must be chosen to adhere to the criterion of equation (7.11). The next stage is to take \( N \) points from the practical output \( y_3(kT) \) and the respective input \( u_q(kT) \). The \( N \) points of \( u(kT) \) and \( u_q(kT) \) need not be drawn from the same instants of time.

With frequency-domain analysis, it is the norm to assume that a sinusoidal input signal has zero phase with respect to the initial time datum. Therefore it is necessary to shift both input waveforms so that they have zero phase with respect to the first sampling instant. In order to maintain fidelity, the output waveforms must be shifted through the same phase angles as their respective inputs.
Considering initially the theoretical sequences $u(kT)$ and $y_i(kT)$, both of these signals are ideal and purely sinusoidal with no additional spectral noise. The first stage is to find the NDFT of $u(kT)$. That is

$$U(\omega, T) = \text{NDFT}(u, \omega, T). \quad (7.15)$$

$U(\omega, T)$ will be a complex value with a magnitude and phase. A sinewave having zero phase with respect to the zero-time datum, has an NDFT which is positive and purely real, from equations (7.13) and (7.14). The shifted NDFT, $U'(\omega, T)$ is therefore

$$U'(\omega, T) = |U(\omega, T)|. \quad (7.16)$$

The shifted input sequence $u'(t)$ can be reconstructed using the inverse NDFT

$$u'(t) = \text{INDFT}(U', \omega, T, N). \quad (7.17)$$

The transition from $U(\omega, T)$ to $U'(\omega, T)$ is achieved by a phase shift $\phi_1$ where

$$U'(\omega, T) = U(\omega, T) e^{i\phi_1} \quad (7.18)$$

and

$$e^{i\phi_1} = \frac{|U(\omega, T)|}{U(\omega, T)}. \quad (7.19)$$

As the input $u(kT)$ has been shifted through an angle of $\phi_1$ radians, the output $y_i(kT)$ will also need to be translated through the same phase shift. Firstly the NDFT of $y_i(kT)$ needs to be found. That is

$$Y_i(\omega, T) = \text{NDFT}(y_i, \omega, T). \quad (7.20)$$

The result needs to be shifted through $\phi_1$ radians, i.e.

$$Y'_i = Y_i(\omega, T) e^{i\phi_1} \quad (7.21)$$

where $e^{i\phi_1}$ is found from equation (7.19).

The output signal can be reconstructed using the inverse NDFT
Digital Filter Error Analysis Using DFT Techniques

\[ y'_1(kT) = \text{INDFT}(Y'_1, \omega, T, N) \] (7.22)

The practical sequences \( u'_q(kT) \) and \( y'_3(kT) \) need to be shifted in order that the input waveform has zero phase with respect to the zero-time datum. However, these sequences contain stochastic noise and hence spectral components other than those at \( \omega \). The NDFT and INDFT together eliminate all frequencies other than \( \omega \), which is necessary for the reconstruction of the deterministic error. In a similar manner to the theoretical data, the practical input \( u'_q(kT) \) is shifted using

\[ u'_q(\omega, T) = \text{NDFT}(u'_q, \omega, T) \] (7.23)

\[ u'_q(\omega, T) = |u'_q(\omega, T)| \] (7.24)

and

\[ u'_q(kT) = \text{INDFT}(U'_q, \omega, T, N). \] (7.25)

\( u'_q(kT) \), a component of \( u'_q(kT) \) contains the fundamental frequency \( \omega \), and has been shifted through \( \phi_2 \) radians to have zero phase shift. Therefore

\[ e^{j\phi_2} = \frac{|u'_q(\omega, T)|}{u'_q(\omega, T)}. \] (7.26)

The component of the practical output, with frequency \( \omega \), after a shift of \( \phi_2 \) radians may be reconstructed as before. That is

\[ Y'_3(\omega, T) = \text{NDFT}(y'_3, \omega, T) \] (7.27)

\[ Y'_3(\omega, T) = Y'_3(\omega, T)e^{j\phi_2} \] (7.28)

and

\[ y'_3(kT) = \text{INDFT}(Y'_3, \omega, T, N). \] (7.29)

The difference between \( y'_1(kT) \) and \( y'_3(kT) \) represents the component of the overall
error at frequency \( \omega \). This comprises the total deterministic error and the component of the stochastic error whose frequency is \( \omega \). If this component of the stochastic error could be removed then the remaining signal will be the deterministic error.

### 7.5.2 Deterministic Error

In order to determine this, the component of the stochastic error at \( \omega \) must first be computed. It is necessary therefore to investigate how the quantisation noise generated by the ADC at frequency \( \omega \) is propagated through the digital filter.

Referring again to Figure 7.2, the discrepancy caused by the quantiser, \( u_{q}(kT) \), is the difference between the ideal input sequence \( u(kT) \) and the practical sequence \( u_{q}(kT) \). The component of the discrepancy at frequency \( \omega \), can be found from the shifted waveforms \( u'(kT) \) and \( u'_{q}(kT) \), viz

\[
u'_{q}(kT) = u'(kT) - u'_{q}(kT).
\] (7.30)

Its NDFT is given by

\[
U'(\omega, T) = NDFT(u'_{q}, \omega, T).
\] (7.31)

Note that \( U'(\omega, T) \) contains just the component of the quantisation noise which has a frequency \( \omega \) and none of the other components. The NDFT of the \( \omega \)-frequency component of the stochastic error \( E_{\omega}(\omega, T) \) that is observed at the output of the digital filter is determined using frequency-domain techniques. So

\[
E'_{\omega}(\omega, T) = U'(\omega, T) H(j\omega T).
\] (7.32)

The reconstructed sinusoidal component of the stochastic error is hence

\[
e'(kT) = INDFT(E'_{\omega}, \omega, T, N).
\] (7.33)

The total error component of angular frequency \( \omega \), comprising both the deterministic and stochastic errors is given by
The deterministic error can be found, as mentioned above, by eliminating the stochastic error from the total error. In other words

\[ e'_{\text{det}}(kT) = y'_1(kT) - y'_3(kT). \]  

(7.34)

7.5.3 Evaluating Stochastic Error

The final stage of the separation technique is to reconstruct the full stochastic error, not just the component at frequency \( \omega \). In order to do this, the original samples of the practical output waveform must be used as opposed to the shifted samples because the shifted samples only contain information relating to the signal at frequency \( \omega \).

To achieve this reconstruction, the ideal shifted waveform \( u'(kT) \) needs to be aligned with the original practical output sequence \( u_q(kT) \). Figure 7.3 shows conceptually how the original input samples might be shifted to the zero phase datum, to align \( u'(kT) \) with \( u_q(kT) \) to give \( u'(kT) \); a phase shift of \( -\phi_2 \) radians (in the opposite direction to the original shift) is needed.

The shifts for the theoretical inputs and outputs can be carried out using

\[ U''(\omega, T) = U'(\omega, T)e^{-j\phi_2} \]  

(7.36) and

\[ Y''(\omega, T) = Y'_1(\omega, T)e^{-j\phi_2}. \]  

(7.37)

The time-domain signals can be reconstructed in the usual way by taking the respective INDFT's, namely

\[ u''(kT) = \text{INDFT}(U'', \omega, T, N) \]  

(7.38) and


**Digital Filter Error Analysis Using DFT Techniques**

\[ y''(kT) = \text{INDFT}(Y''_1, \omega, T, N). \]  
(7.39)

The unshifted total error \( e_{\text{tot}}(kT) \), containing all the spectral components is given by

\[ e_{\text{tot}}(kT) = y''_1(kT) - y_3(kT). \]  
(7.40)

The stochastic error can now be obtained by removing the deterministic error from the total error. As the ideal shifted input was shifted by a further \(-\phi_2\) radians then the deterministic error will also need to be shifted by the same amount. So let

\[ E_d(\omega, T) = E'_d(\omega, T)e^{-j\phi_2} \]  
(7.41)

then

\[ e_d(kT) = \text{INDFT}(E_d, \omega, T, N). \]  
(7.42)

The stochastic error is simply the difference between the total error \( e_{\text{tot}}(kT) \) and the deterministic error \( e_d(kT) \), i.e.

\[ e_d(kT) = e_{\text{tot}}(kT) - e_d(kT). \]  
(7.43)

**7.6 Examples**

Consider the second order continuous transfer function

\[ F(s) = \frac{10s + 10}{s^2 + s + 10}. \]  
(7.44)

Suppose this transfer function is to be realised digitally using the bilinear transform and a sampling interval of \( T=0.01 \) seconds. Suppose also that the coefficients of the practical digital filter are represented to 10 fractional bits. Let the input be a sinewave with a one Volt amplitude, with angular frequency \( \omega=5.3\pi \) radians per second say, and that the practical digital filter contains a quantiser whose resolution is 0.01 Volts. Simulations demonstrating the ideal behaviour of \( F(s) \) and the practical
behaviour of $H(z)$ were performed. Taking care to fulfil the criterion of equation (7.11), a sample of $N=1000$ points of input and output were arbitrarily taken from the theoretical simulation, commencing after 30 seconds. A sample of 1000 points taken from the practical simulation of $u_q$ and $y_3$ was arbitrarily chosen to commence at 37.56 seconds. The analysis outlined in Section 7.5 was performed on the samples and Figure 7.4 illustrates the reconstructed stochastic error and the discrepancy between the actual and reconstructed errors. Figure 7.5 illustrates the reconstructed deterministic error and the associated discrepancy. The 'actual' values were determined using a simulation which modelled the system depicted in Figure 7.2.

Now consider the simple first-order low pass filter

$$F(s) = \frac{1}{s + 1}. \quad (7.45)$$

Using the bilinear transform and a sampling interval of $T=0.01$ seconds as before the digital equivalent may be emulated. As in the previous example, suppose the coefficients of the digital filter are quantised to 10 fractional bits and the quantisation level of the input ADC is 0.01 volts. Let this first order filter be subjected to the same input as the filter in the first example and the samples of the necessary waveforms be taken at the same instants as before. Figure 7.6 illustrates the reconstruction of the stochastic error and from this the magnitude of the discrepancy is negligible.

Suppose now the same analysis is carried out on a similar simple first-order low pass filter, but with a transfer function

$$F(s) = \frac{1}{s + 0.1}. \quad (7.46)$$

Figure 7.7 illustrates the reconstruction of the stochastic error from such an analysis. From this figure it is clear that there is a significant discrepancy between the actual error and the reconstructed error.
This large discrepancy is caused by virtue of the fact that the transients have not fully decayed. For this example, the time constant of the analogue filter is 10 seconds, whilst in the previous example, which gave accurate results, the time constant is just 1 second. Therefore it is imperative to ensure that all transient effects have decayed to a negligible amount before DFT error analysis can successfully be carried out.

The final example to consider is a practical illustration. A digital filter was realised, using the bilinear transform, sampling 100 times per second, to emulate the transfer function of equation (7.45). The coefficients of the digital filter were stored to eight fractional bits and the quantisation level of the analogue-to-digital converter was 0.04 Volts. The input to the digital filter was a sinusoid with an amplitude of about 1 Volt with a frequency of about 2.65 Hz (5.3π radians per second). The signal drove the filter for seventy seconds and the final 1000 samples of the input and output were stored for analysis. Figures 7.8 and 7.9 respectively show the reconstructed deterministic and stochastic errors which were recovered from the stored samples. It can be seen that the reconstruction of the deterministic error is fairly accurate whilst the stochastic error has a large sinusoidal component superimposed upon it. This discrepancy arises due to experimental error in the settings of the amplitude and frequency of the incoming sinusoidal signal. Despite this the practical analysis has validated the technique of extracting the deterministic and stochastic error components from a set of input and output signal samples.

7.7. Summary

A normalised discrete Fourier transform technique has been developed which is more advantageous to use than the accumulative types that are available in some simulation packages such as MATLAB.

It is now possible using spectral techniques to analyze the deterministic and stochastic error forms that are generated by digital filters. This greatly assists in the verification
of error analysis methods that are presented in many control texts. This method of error analysis can play an important role for the design of digital filters to ensure that the magnitudes of the stochastic and deterministic errors are consistent with one another; changes to the filter parameters can be made to ensure that no particular error form dominates [Fo91]. For instance, if the stochastic error is significantly greater than the deterministic error then an ADC with a longer wordlength should be employed in order to reduce the value of $e_q$. If however the deterministic error is the dominant cause of inaccuracy then a change in the value of the sampling interval, $T$, could result in the reduction of the deterministic error [Ol951].

To ensure a reasonable degree of accuracy, it is essential to perform error analysis over a duration of time when all the transient effects can be considered negligible. In this analysis, stochastic error results purely from ADC quantisation, $e_q$, which is a realistic situation when the filter is implemented using a high-level programming language.

![Figure 7.1. The practical system](image)
Figure 7.2. The test system

Figure 7.3. Aligning the theoretical input.
Digital Filter Error Analysis Using DFT Techniques

Figure 7.4. Validation of the $e_s$ reconstruction.

Figure 7.5. Validation of the $e_d$ reconstruction.
Digital Filter Error Analysis Using DFT Techniques

Figure 7.6. Reconstruction of stochastic error for \( F(s) = \frac{1}{s + 1} \).

Figure 7.7. Reconstruction of stochastic error for \( F(s) = \frac{1}{s + 0.1} \).
Figure 7.8. Reconstruction of deterministic error for a practical digital filtering example.

Figure 7.9. Reconstruction of stochastic error for a practical digital filtering example.
8.1 Introduction

In many practical circumstances, no great consideration is given to the selection of a precise value for the sampling frequency of a digital filter. The analyses which follow in this chapter demonstrate how the performance of a digital filter can be greatly improved by careful selection of an appropriate sampling interval.

The chapter investigates how both algorithmic error and coefficient representation error vary with the value of the sampling interval. The two error forms can then be combined so that a value for the $T$ may be chosen which reduces the overall error generated by the filter. Provided that care is taken with the internal arithmetic and structure of the filter, multiple-word truncation error may be reduced to negligible proportions [Fo91].

8.2. Quantised-coefficient algorithmic error

Algorithmic error is defined as the difference between the output of a continuous filter and the output of its ideal discrete equivalent, evaluated at each instant of sampling. In the determination of algorithmic error, infinite precision is assumed, and quantisation
effects do not affect the outcome. Suppose there exists a continuous transfer function \( F(s) \) and an equivalent ideal digital transfer function \( G(z) \) with a sampling interval of \( T \) seconds. Using frequency response analysis, the magnitude of the algorithmic error in response to a unit sinusoid of angular frequency \( \omega \) is given by

\[
| E_a | = | F(j\omega) - G(j\omega T) |. \tag{8.1}
\]

Coefficient representation error exists in digital filters as a result of specifying the filter coefficients to some fixed number of bits. Suppose \( G(z) \) and \( H(z) \) respectively represent ideal and practical digital filters, then by using frequency-domain techniques the peak value for the coefficient representation error is given by

\[
| E_c | = | G(j\omega T) - H(j\omega T) |. \tag{8.2}
\]

In practical circumstances, unless the coefficients of a digital filter with quantised coefficients exactly match those of the ideal filter, algorithmic error cannot be directly measured in isolation from the coefficient representation error. However, it is physically possible to determine the difference between the output of a continuous filter and the output of a practical digital filter at each sampling interval. 'Quantised-coefficient algorithmic error', \( E_d \) (deterministic error), consists of two components; algorithmic error and coefficient representation error and is calculated simply as the sum of the two components. Therefore, for frequency-domain analysis,

\[
| E_d | = | E_a + E_c |
\]

\[
= | F(j\omega) - H(j\omega T) |. \tag{8.3}
\]

8.2.1. Example

Consider the simple compensator whose continuous transfer function is

\[
F(s) = 0.65 \left[ \frac{0.167s + 1}{0.4s + 1} \right] \tag{8.4}
\]

Figure 8.1 shows three curves: pure algorithmic error, \( E_a \), coefficient representation error, \( E_c \) and quantised-coefficient algorithmic error, \( E_d \) for filters emulated from the
Improving the performance of digital filters

first-order transfer function of equation (8.4). The coefficients of the practical digital filter have been represented to 12 fractional bits. These curves have been derived assuming a sinusoidal input signal having an input frequency of 10 radians per second. From the diagram, it is clear that for slow sampling rates, algorithmic error is the dominant form of error whereas for faster sampling rates, coefficient representation error is the predominant error mechanism. From this observation it is apparent that designing a digital filter with the fastest sampling rate that a processor could handle is not necessarily a good idea. By making trade-offs between algorithmic error and coefficient representation error, a compromise can be achieved which could optimise the overall performance of the digital filter. Intuitively the intersection of the algorithmic error and coefficient representation error curves yields a region, called the sampling window, where the best sampling frequencies for the digital filter should lie (see Section 8.3).

The plot of quantised-coefficient algorithmic error, $E_d$, reflects these trends. For relatively fast and relatively slow sampling rates, $E_d$ is large but it is at its least over a range of intermediate sampling frequencies.

8.3. Using a figure of merit to optimise filter performance

Examination of curves of $E_d$ quickly reveals certain sampling frequencies which generate greatly reduced amounts of error. This approach could be useful if the signal frequency is fixed, but this is rarely the case in digital filtering situations - especially digital control applications. For such applications the digital filter would typically be subjected to waveforms consisting of a wide spectrum of sinusoidal components. The figure of merit is a novel concept which arbitrarily describes the overall performance of an emulated digital filter when it is subjected to a spectrum of frequencies, as opposed to a single-component signal.

To determine the figure of merit, the digital filter $H(z)$ should be subjected to a range
of signal frequencies. The range should commence at a suitably low frequency that could be approximated to d.c., and should extend to a suitably high frequency, e.g. the bandwidth of the system, or the maximum required operating frequency, provided that the Nyquist limit is observed. This interval of signal frequencies can now be divided into \( n \) components, \( \omega_1, \omega_2, \ldots, \omega_n \) distributed at equal intervals on a logarithmic scale as shown in Fig. 8.2. The greater the number of points used in the analysis the more confident one can be of the figure of merit. A typical value for \( n \) would be 50, which would take a few seconds to calculate on a personal computer.

8.3.1. Type I figure of merit

A suitable figure of merit, defined as Type I, has been developed to describe the discrepancies in both the gain and phase of the continuous and discrete systems. Figure 8.2 shows typical frequency responses of a continuous filter, \( F(s) \), and a practical digital filter, \( H(z) \), in the gain-phase plane. For each signal frequency, \( \omega_i \) in the gain-phase plane, the distance \( \lambda_i \) between the two frequency points can be computed. The Type I figure of merit can then be found from the average value of \( \lambda_i \).

Suppose the difference in magnitude between the continuous and discrete responses for the \( i \)th signal frequency is

\[
\Delta M_i = \log_{10}|F(j\omega_i)| - \log_{10}|H(j\omega_i,T)|
\]  

and the phase difference, measured in radians, is

\[
\Delta \phi_i = \angle F(j\omega_i) - \angle H(j\omega_i,T)
\]  

then the perpendicular distance, \( \lambda_i \), between the two plots at frequency \( \omega_i \) is

\[
\lambda_i = \sqrt{(\Delta M_i)^2 + (\Delta \phi_i)^2}
\]  

Finally, the figure of merit \( \mathcal{F}_I(T) \) is calculated using
Improving the performance of digital filters

\[ \mathcal{F}_1(T) = -\log_{10} \left( \sum_{i=1}^{n} \frac{\lambda_i}{n} \right) \]  

(8.8)

(This definition for the figure of merit gives an improvement over the original definition pioneered in [Ol95a].)

8.3.2. The Choice of T

Figure of merit calculations can be used to find a value for the sampling interval which can greatly improve the performance of emulated digital filters. The figure of merit calculations are computationally very expensive so it is necessary to have an idea where the best sampling frequencies are likely to lie. The simplest method of achieving this is to plot the curves of algorithmic error and coefficient representation error against sampling frequency, as in Figure 8.1, for an input signal whose frequency equals the maximum frequency that the filter is likely to handle, i.e. system bandwidth. The region around where the two curves intersect provides the range, called the sampling window, in which the optimum sampling frequency should lie, a region where neither algorithmic error nor coefficient representation error dominates.

From this observation the minimum and maximum values of T can be chosen. Commencing at the minimum value for the sampling interval, the figure of merit is calculated for successive values of T, the increment \( \Delta T \) being that permitted by the resolution of the filter's hardware timer. The value of T which produces the greatest value for the figure of merit should be used as the sampling interval for the digital filter.

8.3.3. Example 1

Referring back to the compensator of equation (8.4), suppose that the maximum input frequency to the filter is 10 radians per second, and a suitably low frequency is 0.01 radians per second. Figure 8.1 illustrates the curves of algorithmic error and coefficient
representation error for the filter at the maximum input signal frequency of 10 radians per second. From this figure, the two curves intersect at various points, and from the crossings it can be observed that a suitable value for the sampling frequency is likely to lie somewhere between 30 Hz (T=0.0333 seconds) and 300 Hz (T=0.00333 seconds); thus defining the sampling window. The coefficients of the resulting digital filter are arbitrarily specified to 12 fractional bits.

Figure of merit calculations can be performed over the interval T=0.003 seconds to T=0.034 seconds; both values being adjusted to the nearest thousandth of a second. Assuming the resolution of the filter timer to be 0.001 seconds, Fig. 8.3 shows how the Type I figure of merit of the digital filter, H(z), varies with the sampling interval. From these curves the sampling interval T_s=0.0180 seconds (point a) yields the greatest figure of merit and should therefore be used for the sampling time of the digital filter.

To examine the effectiveness of the figure of merit calculations two simple trials are conducted. The sampling interval of T_c=0.0230 seconds (point b) gives much slower sampling as well as a slightly inferior figure of merit when compared with the chosen value. The sampling interval of T_c = 0.0280 seconds (point c) provides the digital filter with the second highest figure of merit. Point d corresponding to a sampling period of T_d = 0.0040 seconds results in a filter having the poorest figure of merit despite sampling at a much higher rate than T_c. Finally point e (T_e = 0.0200 seconds) lies between point a and point b but has a relatively poor figure of merit.

Subjecting digital filters with sampling times of T_s, T_b, T_c, T_d, and T_e to a unit step input reveals that the filter with the highest figure of merit produces the least error, as demonstrated in Figure 8.4. The filters can also be subjected to a somewhat more complicated waveform consisting of several sinusoidal components, as shown in Figure 8.5. Figure 8.6 (which shows the absolute error on a logarithmic scale to highlight discrepancies) illustrates that the filter with the highest figure of merit again handles the input signal most effectively. In both cases it can be seen that in general the size of the error decreases as the figure of merit increases, as predicted.
8.3.4. Example 2

Consider a continuous notch filter with transfer function

\[ F(s) = \frac{0.0253s^2 + 1}{0.0253s^2 + 0.0159s + 1}. \]  

(8.9)

An example in [Fo91] uses coefficient sensitivity analysis to determine coefficient accuracy requirements. The aim in the example was to achieve 2% accuracy in the continuous numerator coefficients and 10% accuracy in the middle denominator coefficient. It was found that the accuracy specification requires the use of 14 fractional bits in the \( a_0, a_2 \), and \( b_2 \) coefficients and 13 fractional bits for the \( a_1 \) and \( b_1 \) digital filter coefficients. Using a sampling period of \( T = 0.01 \) seconds and the above wordlength requirements, the transfer function of the equivalent digital filter realised using the bilinear transform is

\[ H_1(z) = \frac{16333 - \frac{16300}{8192}z^{-1} + \frac{16333}{16384}z^{-2}}{1 - \frac{16300}{8192}z^{-1} + \frac{16281}{16384}z^{-2}}. \]  

(8.10)

The method of determining coefficient wordlength requirements outlined in [Fo91] is a direct and straightforward procedure. However, can a digital filter having a better performance (or lower coefficient wordlength requirements) be obtained using the heuristic figure of merit approach?

Consider the notch filter of equation (8.9). Using a range of sampling frequencies lying between 10 and 1000 samples per second, Figure 8.7 shows the curves of algorithmic error, coefficient representation error and quantised-coefficient algorithmic error, for digital filters emulated from equation (8.9). The curves have been plotted for a sinusoidal input signal of 10 radians per second and the digital filter coefficients have been represented to just 8 fractional bits. From Figure 8.7, a suitable sampling window will give a range of sampling times varying from 0.0090 seconds to 0.0780 seconds. (Note that the original value of \( T = 0.01 \) seconds lies inside this window.) Assuming
again that the resolution of the timer is 0.001 seconds, Figure 8.8 shows how the Type I figure of merit of the resulting digital filter varies with the sampling interval. Point a, corresponding to $T_a = 0.0390$ seconds, gives the highest figure of merit and should be chosen as the best sampling time. Point b, corresponding to a sampling period of $T_b = 0.0140$ seconds yields a digital filter having a reasonably high figure of merit.

Figure 8.9 shows how the quantised-coefficient algorithmic error varies with time in response to the composite input signal of Figure 8.5 for the original digital filter of equation (8.10) and the two digital filters with coefficients specified to just 8 fractional bits. From the graphs it can be seen that the digital filter realised with a sampling period $T_a$ has a significantly reduced error response to that of $T_b$; this is correctly predicted by the figure of merit curves. Both digital filters realised using coefficients of 8 fractional bits provide superior performance to the example filter realised with a combination of 13- and 14-fractional bit coefficients, emphasizing the benefit of using the figure of merit technique.

Point c, which lies on the figure of merit curve of Figure 8.8 corresponds to a digital filter whose sampling period is $T_c = 0.0100$ seconds, with coefficients represented to just 8 fractional bits. The transfer function of this practical digital filter is

$$H_c(z) = \frac{255}{256} - \frac{509}{256} z^{-1} + \frac{255}{256} z^{-2}$$

Figure 8.10 shows how the errors generated by the digital filters of equation (8.10) and (8.11) vary in response to a composite input. As both filters have identical sampling rates, the digital filter with the longer coefficient wordlength ought to generate the smaller error, but this it is not the case. Surprisingly the filter with the shorter wordlength gives slightly better performance. This can be explained by examining the coefficients of transfer functions (8.9) to (8.11). The dc gain of both the continuous filter and the 8-bit digital filter is unity. However the dc gain of the original digital filter is 1.0154 which leads to degraded performance; the extra five or six bits have
resulted in coefficients which cause the steady-state gain to differ from that intended.

This example has highlighted the value of applying figure of merit analysis to choose the sampling period of a digital filter. If the sampling interval of the original digital filter (having 13 and 14 fractional bit coefficients) had been selected using a figure of merit technique, the resulting digital filter could have, potentially, a very accurate performance.

8.3.5. An alternative figure of merit (Type II)

The Type I definition for a figure of merit is used to predict the overall long-term performance of a digital filter to a general input signal. Such a definition could be beneficial for predicting the behaviour of regulator-type controllers. If the short-term (i.e. transient) performance of a filter to a specific input signal is required then a second figure of merit has been found to be useful.

Unlike the previous definition which is based upon differences between the frequency responses of the practical digital filter and the analogue equivalent, the Type II definition is determined from discrepancies in the time-domain responses. This makes the Type II figure of merit suitable for predicting the performance of servo-type controllers.

Suppose a signal \( u(t) \) drives the input of an analogue filter \( F(s) \) and a practical digital filter \( H(z) \). The output of the continuous filter, when expressed as a Laplace transform is

\[
y_{\text{cont}}(s) = u(s)F(s) \tag{8.12}
\]

and the output of the digital filter is

\[
y_{\text{dig}}(z) = u(z)H(z) \tag{8.13}
\]

when expressed as a z-transform.
Let \( \zeta_i \) represent the difference between the continuous and discrete output values at time \( t = iT \) seconds as shown in Figure 8.11:

\[
\zeta_i = y_{\text{cont}}(iT) - y_{\text{dig}}(iT).
\] (8.14)

For \( n \) sample points, ranging from time \( t = 0 \) seconds to \( t = (n-1)T = t_{\text{max}} \) seconds, the Type II figure of merit is calculated from the average absolute value of \( \zeta_i \), viz

\[
\mathcal{F}_2(T) = -\log_{10} \left( \frac{\sum_{i=1}^{n} |\zeta_i|}{n} \right)
\] (8.15)

### 8.3.6. Example 3

Consider the lightly-damped analogue filter

\[
F(s) = \frac{10}{s^2 + s + 10}.
\] (8.16)

Using coefficients consisting of 8 fractional bits, Figure 8.12 shows the three error curves for sampling rates between 10 and 1000 Hz and the corresponding sampling window. The sampling window for this example gives possible sampling times varying from 0.0180 seconds to 0.0720 seconds. Assuming the digital timer can be adjusted to the nearest thousandth of a second then Figures 8.13(a) and 8.13(b) respectively show how the Type I and Type II figures of merit vary for the practical digital emulations of equation (8.16). (The Type II figure of merit was carried out for a step input over a ten second duration.)

Both figure of merit curves follow a very similar profile; a peak on the Type I curve corresponds, in general, to a peak on the Type II curve. However, consider two sampling periods, \( T_a = 0.0400 \) seconds and \( T_b = 0.0200 \) seconds. The sampling period \( T_a \) results in a digital filter having the highest figure of merit according to the Type I definition, while \( T_b \) produces a digital filter having the greatest Type II figure of merit. This leads to the question: 'Which sampling period should be used - \( T_a \) or \( T_b \)?'
Suppose two digital filters \( H_a(z) \) and \( H_b(z) \) with sampling times of \( T_a \) and \( T_b \) respectively and coefficients of 8 fractional bits, emulate the continuous transfer function of equation (8.16), and both filters are subjected to the composite input signal of Figure 8.5. Figure 8.14 illustrates the discrepancy between the output of the continuous filter and each of the two digital filters at every sampling instant over a twenty second duration. Considering the first ten seconds of simulation: \( H_a(z) \) has a very poor transient response when compared with that of \( H_b(z) \). Considering now the last ten seconds of simulation: in the residual state \( H_a(z) \) gives a much lower error than \( H_b(z) \).

If a value of \( T \) is required to improve the performance of a digital filter (eg for a servo system), to a strictly defined input (eg a unit step), then the Type II figure of merit should be used, leading to the choice of \( T_b \). However, the value \( T_a \), chosen from the type I figure of merit, should be used if the long term, or residual error requires minimisation (eg for a regulator system).

### 8.3.7. Example 4

The aim of this example is to apply the figure of merit technique to improve the performance of a practical digital filter which emulates the continuous filter of equation (8.16).

For this example the input signal is chosen to consist of the same composite waveform of Figure 8.5 along with discontinuities at times \( t = 1 \) and \( t = 4 \) seconds. Figure 8.15(a) illustrates the simulated input signal and Figure 8.15(b) shows the simulated continuous output signal.

Using coefficients specified to 8 fractional bits and using the Type I figure of merit, the sampling period which gives the best filter performance is \( T = 0.0400 \) seconds as in Example 3. The sampling time \( T = 0.0380 \) seconds gives faster sampling but the figure of merit curves predict inferior performance. Table 8.1 summarises the
Coefficient values for both filters in which only $b_1$ is different.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>$T = 0.0400$ seconds</th>
<th>$T = 0.0380$ seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>0.00390625</td>
<td>0.00390625</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.0078125</td>
<td>0.0078125</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.00390625</td>
<td>0.00390625</td>
</tr>
<tr>
<td>$b_1$</td>
<td>-1.9453125</td>
<td>-1.94921875</td>
</tr>
<tr>
<td>$b_2$</td>
<td>0.9609375</td>
<td>0.9609375</td>
</tr>
</tbody>
</table>

Table 8.1. Coefficient values for the $z$-filter emulation of equation (8.16)

Both filters were exposed to the same composite input; the oscilloscope printout of this signal is shown in Figure 8.17. Figure 8.18 shows the oscilloscope printout of the continuous and discrete responses to this signal for the digital filter whose sampling period is 0.0400 seconds. In this demonstration the discrepancy between the continuous and discrete responses is small as the figure of merit analysis would suggest. The equivalent responses for $T = 0.0380$ seconds are shown in Figure 8.19. Despite the faster sampling rate, the discrepancy between the continuous and digital responses is very large, confirming the low figure of merit for that particular sampling period. (The design of the continuous reference filter, which has an approximate transfer function of equation (8.16), is discussed in Chapter 6.)

8.3.8. Example 5

In the previous example the sampling window covered a range of slow sampling frequencies. In some cases it may be necessary to sample faster to further reduce the error. Two ways in which the sampling window can be shifted to a faster sampling region are:

(i) use longer coefficient wordlengths, or

(ii) use the delta operator.

Both techniques effectively reduce the coefficient representation error thus forcing the
Improving the performance of digital filters

intersection of the coefficient representation error curve with the algorithmic error curve to a higher sampling region. This example will endeavour to find higher sampling rates for digital filters emulating equation (8.16), without compromising filter performance, using the delta operator. Throughout this example, the modified canonic structure for the delta operator, developed by Goodall [Go89, Fo91, Go93], will be used having a transfer function:

\[
H(\delta) = \frac{p + qd_1\delta^{-1} + rd_2\delta^{-2}}{1 + d_1\delta^{-1} + d_2\delta^{-2}}. \tag{8.17}
\]

Using coefficients specified to 8 fractional bits, Figures 8.20(a) and 8.20(b) respectively show the error curves and the figure of merit curve for a digital filter, realised using the modified canonic delta form, to emulate the filter of equation (8.16). It is apparent from Figure 8.20(a) that the transition from \( z \) to \( \delta \) gives a sampling window containing faster sampling rates than those of an equivalent \( z \)-filter. Point a, corresponding to a sampling period of \( T_a = 0.0280 \) seconds, gives a digital filter having the highest figure of merit and should give the best filter performance. Point b, situated at \( T_b = 0.0090 \) seconds, gives much faster sampling but a relatively poor figure of merit.

Further improvements in filter performance using higher sampling rates can be achieved if the coefficients are stored in floating-point format. Figures 8.21(a) and 8.21(b) respectively show the errors and figure of merits for modified canonic filters which use floating-point coefficients having 8 bit mantissas. From Figure 8.21(a) it can be seen that faster sampling, without compromising the filter performance, can be obtained using the modified canonic form with floating-point coefficients. The figure of merit analysis reveals that the required sampling period is \( T = 0.0110 \) seconds. Table 8.2 summarises the coefficient values required for both the fixed-point filters and the floating-point modified canonic delta filter.
Improving the performance of digital filters

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>( T_a = 0.0280s )</th>
<th>( T_b = 0.0090s )</th>
<th>( T = 0.0110s ) (floating-point)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>0</td>
<td>0</td>
<td>( 157 / 2^{19} )</td>
</tr>
<tr>
<td>( q )</td>
<td>0.21875</td>
<td>0.08203125</td>
<td>( 202 / 2^{11} )</td>
</tr>
<tr>
<td>( r )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( d_1 )</td>
<td>0.03515625</td>
<td>0.0078125</td>
<td>( 199 / 2^{14} )</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>0.21875</td>
<td>0.08203125</td>
<td>( 203 / 2^{11} )</td>
</tr>
</tbody>
</table>

Table 8.2. Coefficient values for the \( \delta \)-filter emulations of equation (8.16)

Figure 8.22 shows the oscilloscope printout of the continuous response and the delta filter response using fixed-point representation with a sampling period of 0.0280 seconds. (Both systems were subjected to the composite input signal of Figure 8.17.) The discrete response maintains close fidelity with the analogue response except for a large deviation which occurs between 6 and 7 seconds. This discrepancy lies with the continuous response due to approximations in the analogue filter transfer function. (Figure 8.15(b) illustrates the expected theoretical continuous filter response.) Figure 8.23 shows the equivalent oscilloscope printout for the delta filter which samples every 0.0090 seconds. It can be seen from this printout that there is a fairly large deviation between the analogue and digital responses confirming the low figure of merit associated with the delta filter sampling at \( T = 0.0090 \) seconds. Finally Figure 8.24 shows the practical responses of the analogue filter and the delta filter realised using the modified canonic structure and floating-point coefficients. The digital response exhibits excellent fidelity with the analogue output except at the region where the continuous response deviates from its theoretical trajectory.
8.4. Internal variable considerations

In order to select a suitable sampling period for a digital controller, this chapter has so far emphasised the need for consistency between the algorithmic error and coefficient representation to maximise the utilisation of the available filter hardware. A similar approach is used to determine the wordlength requirements of a digital filter.

The internal variables of a digital filter comprise three components: the basic wordlength, some overflow allowance and some underflow allowance. This section details techniques for determining the number of bits for the basic wordlength along with the underflow and overflow allowance.

8.4.1. Determining the basic wordlength

The technique of determining the basic wordlength of the digital filter leads to both the input quantisation step size and the required wordlengths for the ADC and DAC. The ADC wordlength should be chosen such that it is not too small to be detrimental to the performance of the system and not too large that any advantage gained will be eliminated by the other error forms; again emphasising the need for consistency.

The first stage in determining the ADC wordlength is to establish a suitable value for the quantisation level, q, such that the stochastic quantisation noise is of similar order to the deterministic error (the combination of algorithmic error and coefficient representation error). For a digital filter $H(z)$ and input quantisation level, q, then the rms error caused by the quantisation noise, after propagating through the system can be estimated using [Fr90]

$$E_{adc} = q \sqrt{\frac{1}{2\pi} \int_{|z|=1} H(z)H(z^{-1}) \frac{dz}{12z}}$$

(8.18)

The rms quantisation noise generated by the DAC is given simply by
The two expressions of (8.18) and (8.19) should be evaluated in terms of q to find which quantisation source causes the most degradation to the output.

Once the worst source of quantisation noise has been identified, the next stage is to determine a suitable value for the quantisation level, q. Firstly the rms value of the theoretical deterministic input signal should be calculated, either from a simulated time-domain response, or by scaling the total theoretical deterministic error $E_d$ value by $1/\sqrt{2}$. The next step is to choose a sensible value for q such that the error generated by the chosen quantisation error source is of a similar order to (or slightly smaller than) the rms deterministic error.

To determine the number of bits required for the ADC and DAC respectively, the potential input and output ranges of the digital filter require determination. This can be achieved through simulation using a package such as MATLAB or Simulink to estimate the maximum values for the input and output signals of the digital filter. Once the maximum voltage $V_{\text{max}}$ for both the ADC and DAC, and the quantisation level, q, have been established, the required number of bits can be computed using

$$N_{\text{ADC,DAC}} = \text{integer} \left\{ \log_2 \left( \frac{2V_{\text{max}}}{q} \right) \right\} + 1 \text{ bits}$$

The inner term of equation (8.20) computes the number of q Volt quantisation steps required to cover the required voltage range, which is doubled to take negative values into account. The remainder of the equation converts the number of steps to a number of bits, rounded up to the nearest bit.

**8.4.2. Overflow requirements**

The allowance of several overflow bits permits the value of the internal variables to exceed the maximum input and output values [Fo91]. The analysis in this subsection
Improving the performance of digital filters

concentrates on determining the overflow requirement for a standard canonic delta filter. The overflow requirements for a z-filter and a modified canonic delta filter will be stated; the full derivation of these expressions may be found in [Fo91, Go89, Go93]. Consider the algorithm employed in a second-order canonic delta filter (see Figure 8.16) at each sampling instant:

\[
\begin{align*}
\nu &= u - r_1 w - r_2 x \\
y &= c_0 \nu + c_1 w + c_2 x \\
x &= x + w \\
w &= w + \nu.
\end{align*}
\]

(8.21)

At steady-state, the internal variables cease to vary, and this only happens when variables \( \nu \) and \( w \) are both zero. When the steady-state condition is met, the output \( y \) is given by the product of \( c_2 \) and \( x \). For a unit step-input, the output of a second-order delta filter at steady state is given by

\[
y_{ss} = \left(1 - \frac{c_2}{r_2}\right) = c_2 x
\]

(8.22)

From equation (8.22) it can be seen that the size of the internal variable \( x \) is \( 1 / r_2 \) times greater than the input at steady state. The number of bits required for the overflow allowance is therefore given by

\[
N_{\text{overflow}} = \text{integer} \left\{ \log_2 \left( \frac{1}{r_2} \right) \right\} + 1 \text{ bits}
\]

(8.23)

If the designed system is known to have a lightly-damped response, an extra bit should also be allowed for overshoot during the transient stage. For a first-order system, similar analysis can be performed and the value \( r_2 \) in equation (8.23) would be replaced by \( r_1 \).

For a z-filter, it can be shown [Fo91] that the overflow requirements are given from the expression
Improving the performance of digital filters

\[ N_{\text{overflow}} = \text{integer} \left\{ \log_2 \left( \frac{1}{1 + b_1 + b_2} \right) \right\} + 1 \text{ bits} \tag{8.24} \]

This formula is applicable for first-order filters and second-order filters having real poles. For second-order filters having complex poles, again, an extra bit should be required to allow for overshoot. By inspection of equations (8.23) and (8.24), the overflow requirements for the variables of a z-filter and a canonic delta filter will be identical.

The beauty of a delta filter realised in the modified canonic form is that the internal variables will be scaled as the input, unlike the previous two configurations. From [Go93], the overflow allowance for a modified canonic delta filter is given by

\[ N_{\text{overflow}} = \text{integer} \left\{ \log_2 \left( \frac{1}{2\xi} \right) \right\} + 1 \text{ bits} \tag{8.25} \]

where \( \xi \) is the damping factor of the filter for complex poles.

\section*{8.4.3. Underflow requirements}

In a digital filter the internal variables are computed from the product of a variable with a coefficient; the wordlength of the resulting variable would be equal to that of the coefficient plus that of the original variable. In recursive digital filters multiple-word truncation is essential to prevent the lengths of the internal variables growing indefinitely. However, it is necessary to make a provision of several underflow bits for the internal variables such that the digital filter responds properly to a 1 LSB change of the input [Fo91]. Provided care is taken to implement a sufficient underflow allowance for the internal variables in a digital filter, multiple-word truncation error can be reduced to negligible proportions. The method laid out in this subsection provides a means of determining the underflow requirements for a z-filter along with the canonic and modified canonic delta structures, but can be readily applied to any filter structure.
Improving the performance of digital filters

Let

\[ \Phi_{adc} = \frac{1}{2\pi j} \oint_{|z|=1} H(z)H(z^{-1}) \frac{dz}{z} \]  

(8.26)

where \( H(z) \) is the transfer function of the digital filter and \( \Phi_{adc} \) relates the variance of the ADC quantisation error to the variance of the quantisation noise. Also let

\[ \Phi_m = \frac{1}{2\pi j} \oint_{|z|=1} H_m(z)H_m(z^{-1}) \frac{dz}{z} \]  

(8.27)

where \( H_m(z) \) is the transfer function relating the source of the multiple-word truncation to the output of the digital filter and \( \Phi_m \) relates the multiple-word truncation error variance to the variance of the quantisation noise.

The square root of the ratio \( \Phi_m : \Phi_{adc} \) indicates the ratio between the rms multiple-word truncation error and the rms ADC quantisation error for identical quantisation levels. In order to reduce the rms multiple-word truncation error to a fixed percentage of the rms ADC quantisation error the required underflow allowance is given by

\[ N_{underflow} = \text{integer} \left\{ \log_2 \left( \frac{100}{\text{acc}} \sqrt{\frac{\Phi_m}{\Phi_{adc}}} \right) \right\} \text{ bits} \]  

(8.28)

where \( \text{acc} \) is the required percentage accuracy.

The required transfer functions relating the multiple-word truncation error to the output, \( H_m \), are tabulated in Table 8.3 for the z-filter, the delta filter, along with the Class I and Class II modified canonic delta structures. In order to evaluate expressions (8.26) and (8.27) for any of the delta filters, the filter should be re-expressed in terms of \( z \). Chapter 4 presents several techniques for evaluating expressions (8.26) and (8.27).

8.5. Software design

The process of selecting a sampling period that ensures that the coefficient representation error and algorithmic error are of similar orders of magnitude to
Improving the performance of digital filters

Optimise the overall filter performance, can be automated in software as an off-line process. Such software can also be used to determine the internal variable requirements for the resulting filter.

Figure 8.25 shows the top-level diagram of the software. From this it can be seen that the first stage of the design is for the user to enter the coefficients of the analogue transfer function.

Using the structure chart of Figure 8.26, the user is prompted to enter relevant calculation data. Assuming the digital filter hardware has a facility for adjusting the sampling frequency, the user is prompted to enter the lowest possible sampling rate permitted by the hardware consistent with the Nyquist limit. The user will then be prompted to enter the maximum sampling rate and details about the coefficient quantisation. A menu is presented to allow the choice of fixed-point or floating-point representations; an option of exact coefficients is also given. The user will be prompted to enter the coefficient wordlength; the number of fractional bits for fixed-point, or the number of mantissa bits for floating-point arithmetic.

Figure 8.27 presents the structure chart showing how the error curves are plotted. The range of sampling frequencies, as defined by the user, is logarithmically distributed over 100 sampling points. For each sampling point, the analogue transfer function $F(s)$ is translated into the $z$-domain using the bilinear transform to give $G(z)$. Using the required quantisation method, the coefficients of $G(z)$ are reduced to a fixed-length to give $H(z)$. The errors introduced in the translation from $F(s)$ to $G(z)$ (algorithmic error), the translation from $G(z)$ to $H(z)$ (coefficient representation error), and the translation from $F(s)$ to $H(z)$ (quantised-coefficient algorithmic error) are determined using frequency domain techniques. Once the errors have been calculated for each sampling point, the graphs of the three errors are plotted against sampling frequency.

From these curves, the user is prompted to define the sampling window where the algorithmic error and the coefficient representation error are consistent with each other.
Improving the performance of digital filters

This process is highlighted in the structure chart of Figure 8.28. The user is prompted to mark the sampling window by selecting a minimum and maximum sampling rate using the mouse. Following this, the user is asked to enter the resolution of the digital timer. Using these parameters the software establishes a vector of sampling times starting at the shortest value, increasing in steps equal to the resolution of the timer, up to the longest possible value of \( T \).

Referring back again to Figure 8.25, the figure of merit is then calculated for each value of sampling time inside the sampling window. For each sampling period, the equivalent digital filter is derived from the analogue using the bilinear transform and the coefficients are then quantised. The figure of merit for the resulting filter is calculated using the Type I definition.

The value of \( T \) which produces the greatest figure of merit is automatically chosen as the sampling period for the digital filter. Using that particular value of \( T \), the digital filter is emulated and its coefficients are quantised. From this point the internal variable requirements are found.

Although no source code has been explicitly supplied in this thesis for the digital filter design software, Appendix E lists software for improving the performance of digital controllers (Chapter 9). This source code may be easily modified to automate the design of digital filters.

8.6. Summary

By assessing algorithmic error and coefficient representation error for a given digital filter, a technique has been developed for locating a particular value for the sampling interval which can significantly improve the performance of filters designed by emulation. Algorithmic error, \( E_a \), arises in the transition from the continuous domain to the discrete domain and is due to the quantisation of time; its value decreases as the sampling rate is increased. Coefficient representation error, \( E_c \), occurs as a result of the
coefficients of an ideal digital filter being specified to finite accuracy. This type of error tends to increase as the sampling rate is raised albeit in a random manner. Trade-offs between these two error forms, over a wide range of sampling rates, quickly lead to a region where an optimum sampling frequency lies. A combination of the two error forms leads to quantised-coefficient algorithmic error, $E_d$, which is the error that can be observed in a practical digital filter.

By assessing the performance of the filter over the relevant frequency range, a figure of merit for the digital filter may be obtained. By comparing figures of merit for digital filters of differing sampling times, a suitable value for the sampling interval can be found which minimises the overall error. This technique may be used to produce filters having a high performance using low cost hardware which could be very valuable in, for example, in the mass production of low cost digital controllers.

Two definitions for the figure of merit of a digital filter have been presented to predict the performance of the digital filter under various input conditions. In some situations particular emphasis may be placed on the behaviour of the filter in response to certain fixed frequencies. If the Type I figure of merit definition is used, the figure of merit can be weighted in favour of such frequencies.

In some circumstances, such as power electronics applications, where the sampling rate of a digital filter is fixed, consideration of figure of merit for a range of coefficient wordlengths may play a crucial role in determining a sensible value for the number of bits required for the coefficients. Using too few bits in the coefficient wordlengths could lead to digital filters having poor responses. Conversely implementing too many bits may be excessive and in some cases lead to reductions in filter performance.

Further considerations of other quantisation effects need also to be investigated. An ADC should be used whose wordlength is such that the ADC quantisation error is consistent with that of the quantised-coefficient algorithmic error. The effects of multiple-word truncation error can be deemed minimal provided that a sensible
structure for the filter is implemented and care is taken with the internal arithmetic, as
detailed in the internal variable analysis.

The use of the delta-operator also leads to further improvements in the filter
performance [Go89] and significant gains in coefficient accuracy can be obtained by
forsaking the z-operator in favour of delta.

<table>
<thead>
<tr>
<th>Filter type</th>
<th>First-order</th>
<th>Second-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>z-filter</td>
<td>$H_m(z) = \frac{a_0 + a_1 z^{-1}}{1 + b_1 z^{-1}}$</td>
<td>$H_m(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$</td>
</tr>
<tr>
<td>standard canonic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>delta-filter</td>
<td>$H_m(\delta) = \frac{c_0 + c_1 \delta^{-1}}{1 + r_1 \delta^{-1}}$</td>
<td>$H_m(\delta) = \frac{c_0 + c_1 \delta^{-1} + c_2 \delta^{-2}}{1 + r_1 \delta^{-1} + r_2 \delta^{-2}}$</td>
</tr>
<tr>
<td>standard canonic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>delta-filter</td>
<td>$H_m(\delta) = \frac{(q-p)\delta^{-1}}{1 + d_1 \delta^{-1}}$</td>
<td>$H_m(\delta) = \frac{(q-p)\delta^{-1} + (rd_1 - pd_2)\delta^{-2}}{1 + d_1 \delta^{-1} + d_2 \delta^{-2}}$</td>
</tr>
<tr>
<td>Class I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modified canonic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>delta-filter</td>
<td>$H_m(\delta) = \frac{q-p}{1 + d_1 \delta^{-1}}$</td>
<td>$H_m(\delta) = \frac{(q-p) + (rd_1 - pd_2)\delta^{-1}}{1 + d_1 \delta^{-1} + d_2 \delta^{-2}}$</td>
</tr>
<tr>
<td>Class II</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modified canonic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.3. Transfer functions relating the prominent source of multiple-word
truncation error to the output.
Improving the performance of digital filters

Figure 8.1. Variation of $E_a$, $E_c$, and $E_d$ for a fixed-frequency input signal using coefficients having 12 fractional bits.

Figure 8.2. Determining the Type I figure of merit for a digital filter.
Improving the performance of digital filters

Figure 8.3. Variation of figure of merit against sampling period for the compensator of equation (8.4).

Figure 8.4. Error responses of the equivalent digital filters of equation (8.4) in response to a unit step input.
Improving the performance of digital filters

Figure 8.5. A composite input waveform consisting of several sinusoidal components.

Figure 8.6. Absolute error generated at each sampling instant by each digital filter from Example 1 in response to the input waveform of Figure 8.5.
Improving the performance of digital filters

Figure 8.7. Variation of $E_a$, $E_c$ and $E_d$ with sampling rate for a sinusoidal input and 8 fractional bit coefficients for equation (8.9).

Figure 8.8. Variation of figure of merit against sampling period for the filter of equation (8.9).
Improving the performance of digital filters

Figure 8.9. Error responses of the equivalent digital filters of equation (8.9) in response to the composite input of Figure 8.5.

Figure 8.10. Error responses of digital filters in response to a composite input using the original digital filter and an equivalent with 8 fractional bit coefficients.
Improving the performance of digital filters

Figure 8.11. Determining the Type II figure of merit for a digital filter.

Figure 8.12. Determining the sampling window for the digital emulation of equation (8.16) with coefficients of 8 fractional bits.
Improving the performance of digital filters

Figure 8.13. Variation of the Type I and II figures of merit against sampling frequency for the filter of equation (8.16).

Figure 8.14. Error responses of the equivalent digital filters of equation (8.16) in response to a composite input of Figure 8.5.
Improving the performance of digital filters

Figure 8.15. Theoretical input and output signals for the continuous system of Example 4.

Figure 8.16. Block diagram of a general second-order delta filter.
Figure 8.17. Oscilloscope printout of the practical input signal.

Figure 8.18. Oscilloscope printout of the practical continuous output and digital output for a z-filter with sampling period, \( T = 0.0400 \) seconds.

Figure 8.19 Oscilloscope printout of the practical continuous output and digital output for a z-filter with sampling period, \( T = 0.0380 \) seconds.
Figure 8.20. Error curves and figure of merit curves for the modified canonic implementation of Example 5 using fixed-point coefficients.

Figure 8.21. Error curves and figure of merit curves for the modified canonic implementation of Example 5 using floating-point coefficients.
Figure 8.22 Oscilloscope printout of the practical continuous output and digital output for a modified canonic $\delta$-filter using fixed-point coefficients with sampling period, $T = 0.0280$ seconds.

Figure 8.23 Oscilloscope printout of the practical continuous output and digital output for a modified canonic $\delta$-filter using fixed-point coefficients with sampling period, $T = 0.0090$ seconds.

Figure 8.24 Oscilloscope printout of the practical continuous output and digital output for a modified canonic $\delta$-filter using floating-point coefficients with sampling period 0.0110 seconds.
Improving the performance of digital filters

Figure 8.25. Top-level structure chart of software to automate the process of improving the performance of a digital filter.

Figure 8.26. Structure chart demonstrating how the calculation data is entered.
Improving the performance of digital filters

Figure 8.27. Structure chart illustrating the process of plotting the error curves.

Figure 8.28. Structure chart showing how the sampling window may be selected.
9.1 Introduction

The application of a digital compensator in feedback applications will always result in a closed-loop control system whose response differs from the equivalent analogue control system. This Chapter explores techniques for improving the performance of digital controllers applied in closed-loop systems with respect to the analogue counterpart. By making trade-offs between the error forms inherent in digital filters, it is possible to design control systems which make the best use of the available hardware.

The first section of the Chapter reviews various digital design techniques based on classical design methods to determine the best method of designing a digital controller. Once a reliable design technique has been established, a figure of merit approach can be applied to select a suitable sampling frequency in which the overall error is minimised for a given coefficient wordlength. An analogue-to-digital converter should then be chosen such that the magnitude of the ADC quantisation error is of similar order to the rest of the errors. The determination of the wordlengths for the internal variables completes the design process.
The technique of improving the performance of digital controllers is applied to several common control schemes. The topologies that are investigated are:

- simple unity feedback control system
- dual feedback control system with 1st derivative feedback, and
- dual feedback control system with plant and controlled actuator.

The chapter presents the design of a software tool which can be applied to aid the design of practical digital controllers according to the specified design rules. The ideas presented are supported using both theoretical simulation and practical results.

### 9.2. Digital Design Techniques

Consider the closed-loop continuous feedback system of Figure 9.1 where \( P(s) \) is the plant model and \( C(s) \) is the continuous compensator. As an example, suppose the plant \( P(s) \) has transfer function

\[
P(s) = \frac{50}{s(s^2 + 11s + 10)}.
\]

In order to achieve adequate closed-loop control, a phase advance compensator is required whose transfer function is

\[
C(s) = \frac{0.58s + 0.70}{0.16s + 1}.
\]

A key issue in designing digital controllers is the stability of the closed-loop system and its relationship to the sampling interval. The input to the physical system is driven by the output signal from the digital controller via a zero-order hold (ZOH). This means that the plant is driven by a sequence of flat-topped pulses which inherently adds phase lag to the system. Figure 9.3 shows the gain-phase plot of the continuous plant of equation (9.1) along with gain-phase plots of the same plant driven from a ZOH for different values of \( T \). As the duration of the sampling time is increased, the width of the flat-topped pulses driving the plant increases. This causes excessive phase lag which reduces the stability of the system and extra care should therefore be taken.
to design digital compensators to overcome this problem. This section reviews the design of digital controllers using classical control techniques and examines their effectiveness. As an example the performance of the closed-loop system defined in equations (9.1) and (9.2) will be the target specification by which the digital control system, sampling ten times per second, will be designed.

9.2.1. Emulation

Digital control design through emulation involves designing the entire control system using classical techniques and directly converting the continuous compensator to the discrete domain using one of the many available emulation techniques. Choosing the bilinear transform due to its popularity and effectiveness the transfer function of the digital compensator is given by

\[ C(z) = \frac{2.9286 - 2.5982z^{-1}}{1 - 0.5238z^{-1}} \]  (9.3)

9.2.2. W-plane design techniques

Phase lag will always be introduced into digital control systems by the mere existence of a zero-order hold, which increasingly destabilises closed-loop systems for longer sampling times. This problem can be reduced by the use of W-planes which are essentially continuous domains to take into account the phase lag introduced by the ZOH. Two definitions of W exist:

9.2.2.1. w-plane design

The notion of the w-plane, [Le85], which is seldom used, can be used to produce digital controllers having adequate performance [Ph70].

In order to design a digital compensator in the w-plane the following procedure can be applied: The first stage is to determine the ZOH model of the plant using

\[ P_z(z) = \frac{z - 1}{z} \left( \frac{P(s)}{s} \right) \]  (9.4)
Improving the performance of digital controllers

(The subscripted \( z \) implies a zero-order hold in the plant model). The next stage is to convert the ZOH model to the w-plane using a reverse bilinear transformation:

\[
\frac{z}{1 - w} = \frac{1 + w}{1 - w} \quad (9.5)
\]

to give \( P(z) \). Using classical control design techniques an analogue compensator \( C(w) \) can be designed, observing that a signal frequency \( \omega \) in the s-plane is

\[
\omega_w = \tan\left(\frac{\omega T}{2}\right) \quad (9.6)
\]

in the w-plane [Le85]. The designed compensator is then translated into the z-domain to give \( C(z) \) using a bilinear transformation:

\[
w = \frac{z - 1}{z + 1} \quad (9.7)
\]

As an example, using a sampling period of \( T = 0.1 \) seconds, a w-plane compensator which compensates the plant of equation (9.1) to give the same crossover frequency and phase margin as the cascaded system of equations (9.1) and (9.2) has the transfer function

\[
C(w) = \frac{11.69w + 0.57}{2.72w + 1} \quad (9.8)
\]

Using (9.7) the resulting discrete transfer function is

\[
C(z) = \frac{3.2903 - 2.9867z^{-1}}{1 - 0.4629z^{-1}} \quad (9.9)
\]

9.2.2.2. \( w' \)-plane design

A second method of \( W \)-plane design involves the use of the \( w' \)-plane [Le85] and the design method is similar to that for the w-plane.

The first stage is to determine the zero-order hold model of the plant, \( P(z) \) using
Improving the performance of digital controllers

equation (9.4). This model is then translated into the \( w' \)-plane using Tustin’s reverse transformation

\[
z = \frac{1 + w'T/2}{1 - w'T/2}
\]  
(9.10)

to give \( P_z(w') \). A compensator \( C(w') \) can be designed using classical control design methods without the need for frequency scaling. The final stage is to discretise \( C(w') \) using Tustin’s transformation

\[
w' = \frac{2}{T} \frac{z - 1}{z + 1}.
\]  
(9.11)

Using a sampling period of \( T = 0.1 \) seconds, the \( w' \)-plane controller designed to compensate the plant of equation (9.1) has the transfer function

\[
C(w') = \frac{0.58w' + 0.56}{0.14w' + 1}.
\]  
(9.12)

The resulting digital controller is

\[
C(z) = \frac{3.2433 - 2.9438z^{-1}}{1 - 0.4677z^{-1}}.
\]  
(9.13)

9.2.3. Comparisons

For the plant of equation (9.1) Figure 9.4 shows closed-loop step responses for the continuously and digitally controlled systems where the controller has been designed using emulation and both \( W \)-plane techniques. The considerations of the phase lag in \( W \)-plane design methods leads to control systems whose step responses have a significantly closer resemblance to the continuous system than the controller designed directly through emulation. The digital controller designed in the \( w' \)-plane produces similar responses to the controller designed in the \( w' \)-plane even though it is rarely used in practice.

One technique for improving the performance of digital controllers, therefore, is to use
one of the $W$-plane design methods. Controller design using the $w'$-plane will be used throughout the remainder of the Chapter due to its popularity and simplicity over $w$-plane design. For notational convenience, the $w'$-plane will be referred to as the $w$-plane for the rest of this chapter.

### 9.3. Improving the performance of single-loop control systems

Consider the single loop analogue-control and digital-control systems of Figure 9.1 and 9.2. This section investigates, using error analysis, how the parameters for the digital controller can be chosen such that the closed-loop performance mimics, as far as possible, that of the analogue system.

#### 9.3.1. Choosing a value for $T$

The error analysis that is used to improve the performance of emulated digital filters can be applied to the optimisation of digital controllers but with some changes in the definitions of the transfer functions.

Referring to Figure 9.5 let $F(s)$ be the transfer function relating the output of the closed-loop continuous control system to the input, viz

$$F(s) = \frac{C(s)P(s)}{1 + C(s)P(s)} \quad (9.14)$$

where $P(s)$ is the transfer function of the plant and $C(s)$ is the transfer function of the compensator.

The discrete transfer function of the closed-loop system where the digital compensator $C(z)$ has ideal coefficients is

$$G(z) = \frac{C(z)P_z(z)}{1 + C(z)P_z(z)} \quad (9.15)$$

where $P_z(z)$ is the zero-order hold model of the plant.
From Figure 9.5, let $H(z)$ be the closed-loop discrete transfer function of the control system using a compensator $C_q(z)$ having quantised coefficients:

$$H(z) = \frac{C_q(z)P_2(z)}{1 + C_q(z)P_2(z)}. \quad (9.16)$$

Suppose each system is exposed to a sinusoidal input of unit amplitude and angular frequency $\omega$, then algorithmic error is given by

$$E_a(\omega, T) = |F(j\omega) - G(j\omega T)|, \quad (9.17)$$

coefficient representation error given by

$$E_c(\omega, T) = |G(j\omega T) - H(j\omega T)|, \quad (9.18)$$

and the deterministic, or total error is given by

$$E_t(\omega, T) = |F(j\omega) - H(j\omega T)|, \quad (9.19)$$

where $T$ is the sampling period of the digital controller.

The above error analysis however only takes into consideration the error generated by a single unit sinusoid of angular frequency $\omega$. In practice the digital controller will be subjected to inputs containing several frequency components. Even though the error analysis detailed previously is not suitable for evaluating the overall performance of the digital controller, it is useful for finding a range of sampling frequencies where the optimum sampling rate is likely to lie. It is therefore necessary to consider the error over a wide range of working frequencies, that is to determine an overall "Figure of Merit" for the system for each value of $T$. Chapter Eight introduced two definitions by which the Figure of Merit can be determined. The Type I definition (see section 8.3.1) is suitable for optimising the performance of regulating systems where accurate residual performance is essential. The Type II definition (see section 8.3.5) can be used to improve the performance of servo systems where fast accurate tracking is required. Using either definition, the greater the figure of merit, the smaller the overall error and the better the performance of the digital control system with respect to the analogue...
As an example, the compensator of equation (9.2) can control the plant of equation
(9.1) in a closed-loop combination denoted $F(s)$. For various values of $T$, a digital
compensator $C(z)$ can be designed using the w-plane technique - the transfer function
of the closed loop system being $G(z)$. The equivalent system with digital filter
coefficients specified to 8 fractional bits is $H(z)$. Figure 9.6 shows how the algorithmic
error, coefficient representation error and total error vary for an input sinusoid of
angular frequency $\omega$, as the sampling rate is increased. Algorithmic error decreases as
the sampling rate is increased, while coefficient representation error increases in a
random manner. The region where the curves of algorithmic error and coefficient
representation error are of similar orders of magnitude result in a region where the total
error is likely to take its lowest value. This region is defined as the 'sampling window'.

Figure 9.7(a) shows how the total deterministic error varies with sampling rate for a
fixed signal frequency $\omega$. It can be seen that sampling too quickly or too slowly results
in excessive error values. Sampling at an intermediate rate (in the sampling window)
leads to an overall reduction of the error for that particular signal frequency. For the
same range of sampling frequencies, Figure 9.7(b) shows how the figure of merit
varies. The figure of merit for the closed-loop digital control system is significantly
higher inside the sampling window than it is outside. However, the peak value for the
figure of merit does not necessarily coincide with the minimum error, as can be seen
from Figure 9.7. This occurs because the input signal to the digital controller is not
composed solely of one signal frequency, but many.

Once the sampling window has been established a range of possible values for the
sampling period is readily determined, incrementing in steps set by the resolution of
the digital timer. For each value of $T$ in the sampling window, a digital controller is
designed using w-plane design to match the performance of the continuous controller
at some predetermined frequency, and the controller coefficients are subjected to
quantisation. The figure of merit is calculated for the resulting digital control system
Improving the performance of digital controllers

for each value of T and the sampling time which produces the highest figure of merit should therefore be chosen as the sampling interval for the digital controller.

For the system of equations (9.1) and (9.2), Figure 9.8 shows the figure of merit for the digital control systems realised with all values of T lying within the sampling window assuming the digital timer can be adjusted in 0.0001 second intervals. The figure of merit is highest for \( T = 0.0122 \) seconds, so this is chosen as the sampling interval for the digital controller.

Figure 9.9 shows the closed-loop step responses of a digital control system to control the plant of equation (9.1). This is done for three values of T:-

(i) \( T = 0.0122 \) seconds (the highest figure of merit - 0.26)
(ii) \( T = 0.0121 \) seconds (a lower figure of merit (0.17) and faster sampling)
and (iii) \( T = 0.0015 \) seconds (the lowest figure of merit - 0.09).

The continuous response is shown for comparison. It can be seen that the controller with the highest figure of merit gives the best closed-loop response with respect to the analogue equivalent.

By choosing a value of T corresponding to the highest figure of merit, the performance of the digitally controlled system can be improved. However it may be economical, in some circumstances, to sacrifice a small amount of controller performance for a slower sampling speed.

9.3.2. Wordlength requirements

Once the sampling period has been selected, the final step is to choose suitable wordlengths for the internal variables. The internal variables consist of three components:

(i) The basic (ADC) wordlength
(ii) An underflow allowance
(iii) An overflow allowance.
Chapter eight presents analyses for determining these requirements.

The ADC wordlength should be chosen such that it is not too small to be detrimental to the performance of the closed-loop system and not too large that any advantage gained will be eliminated by the other error forms. By treating the w-plane compensator $C(w)$ and the equivalent practical digital compensator $C_q(z)$ as filters, the quantisation level for the ADC can be chosen such that the ADC quantisation error is of similar magnitude to the deterministic error introduced in the transition from $C(w)$ to $C_q(w)$.

Unfortunately stochastic noise analysis always reveals infinite error for integrating transfer functions. In order to determine the required ADC or DAC quantisation level for an integrating controller (such as a P+I compensator), Slaughter's method [SI64] provides a means of determining the worst-case steady-state bound on the output error with respect to the quantisation level.

**9.4. Software Design**

This section details the design of software that can be used to determine the best parameters for a digital controller. The top-level structure chart of the software is illustrated in Figure 9.10.

The first stage is to allow the user to enter the plant transfer function by entering the numerator coefficients and the denominator coefficients. The next stage is to design a reference analogue controller to adequately control the plant in a closed-loop system. The resulting closed-loop system will be the reference against which the digital control system can be designed. From Figure 9.10 the user has a choice of two options of how the analogue controller can be designed. The first choice is to enter the numerator and denominator coefficients of an existing analogue controller. The second choice is to use the continuous controller design software of Appendix D.
Following this the user is prompted to enter parameters relating to the digital control system as shown in Figure 9.11. The user is prompted to enter the minimum and maximum possible values for the sampling rate of the controller. The minimum value is determined by the Nyquist limit of the system and the maximum value is determined from the available hardware. The user is then prompted to enter details about the coefficient quantisation and enter the appropriate value for the coefficient wordlength. The quantisation methods available are fixed-point, floating-point and no quantisation.

A set of digital controllers is automatically designed for the range of sampling frequencies bounded by the minimum and maximum sampling rates as shown in Figure 9.12. For each value of T in the sampling range, the closed-loop transfer function of the analogue control system is determined as a reference. The w-plane model of the plant is determined for the appropriate value of T. A continuous controller is designed in the w-plane to have the same crossover frequency and phase margin as the original analogue system. Using the bilinear transform the w-plane compensator is translated to the discrete domain and the coefficients are quantised accordingly. For each value of T, the errors caused by each digital controller are calculated. To achieve this the frequency responses of the closed-loop continuous system and the closed-loop digital control systems with ideal quantised coefficients are determined. From the frequency responses, taken at one-tenth of the crossover frequency (This was found, through trial and error to be a reliable frequency where the figure of merit mirrored the deterministic error), the algorithmic error, the coefficient representation error and the total deterministic error are readily calculated.

The plots of algorithmic error, coefficient representation error and the total deterministic error are then plotted on the same set of axes.

The user is prompted to select a suitable range of sampling frequencies where the algorithmic error and coefficient representation error are of similar orders of magnitude. Figure 9.13 shows the structure chart for this piece of software. The user selects a point on the error plots corresponding to a suitably low value for the sampling
Improving the performance of digital controllers

frequency and then a point for a suitably high sampling frequency. The user will then be prompted for the step size, or resolution, of the digital timer. From this a vector comprising a set of possible sampling periods is determined.

For each value of T in this vector, the figure of merit for the resulting digital controller is calculated, as shown in the structure chart of Figure 9.14. Using the w-plane technique, the digital controller is designed and the coefficients quantised appropriately. The discrete closed-loop transfer function of the closed-loop digital control system is determined. This is used against the reference analogue control system to calculate the Type II figure of merit for the digital controller.

Figure 9.15 shows the structure chart for determining the best sampling period. A sorting algorithm is used to determine which value of T gives the highest figure of merit. Using this sampling period, the appropriate digital controller is created using the w-plane technique and the coefficients are quantised.

Once the digital controller sampling period and coefficients have been established, the next stage is to determine the ADC and DAC requirements, as shown in Figure 9.16. The first step is to determine the value of the quantisation level, q. This is achieved by plotting the rms total deterministic error against sampling frequency for the digital controller with respect to the w-plane controller. For several possible quantisation levels, the estimated rms ADC quantisation error is also plotted. The user should then select a value for q which produces ADC quantisation error which is consistent with the deterministic error. The ADC and DAC wordlength requirements are found using the quantisation step size along with the maximum controller input and output values.

The last stage of the digital control design process is the determination of the internal variable requirements. Figure 9.17 shows that there are three steps to this. Firstly, the basic wordlength is found from the ADC wordlength. The second step is to compute the necessary overflow requirements and the final stage is to determine the underflow requirement.
9.5. Practical Examples

In order to verify in practice the process of designing a digital control system using the digital control design software which is described above, two practical examples have been carried out. These examples emphasize the importance of choosing a digital controller possessing a high figure of merit.

9.5.1 Examples 1 & 2

Consider the systems of Figure 9.18 (Example 1) and Figure 9.19 (Example 2). A digital controller is required for each example to replace the analogue equivalent. The coefficients of the digital controller will be specified to just four fractional bits to highlight deviations from the optimum.

Figures 9.20 and 9.21 illustrate the error curves and figures of merit for Example 1 and Example 2 respectively. For Example 1, the figure of merit is greatest for $T = 0.0120$ seconds and much lower for $T = 0.0130$ seconds. For Example 2, the maximum figure of merit occurs at $T = 0.0180$ seconds. The figure of merit is significantly reduced for the faster sampling time of $T = 0.0150$ seconds.

For Example 1, the required digital controller is

\[ C_q(z) = \frac{3.8750 - 3.6875z^{-1}}{1 - z^{-1}} \]  

(9.20)

for $T = 0.0120$ seconds or

\[ C_q(z) = \frac{3.9375 - 3.6875z^{-1}}{1 - z^{-1}} \]  

(9.21)

for $T = 0.0130$ seconds.

Figure 9.22 shows the theoretical step responses of the closed-loop analogue control system and the two digital control systems. The system with the highest figure of merit ($T = 0.0120$ seconds) provides a step response closest to the analogue equivalent.
Improving the performance of digital controllers

Figure 9.23 illustrates the practical closed loop step responses for the analogue control system and the digital control system for $T = 0.0120$ seconds, whilst Figure 9.24 shows the equivalent responses for the digital control system with $T = 0.0130$ seconds.

For Example 2, the required digital controller is

$$C_q(z) = \frac{5.9375 - 5.6250z^{-1}}{1 - 0.8125z^{-1}}$$  \hspace{1cm} (9.22)

for the system with the highest figure of merit, $T = 0.0180$ seconds, or

$$C_q(z) = \frac{5.9375 - 5.6875z^{-1}}{1 - 0.8750z^{-1}}$$  \hspace{1cm} (9.23)

for the faster sampling period of $T = 0.0120$ seconds.

Figure 9.25 shows the theoretical step responses of the closed-loop analogue and digital control systems. Figure 9.26 and Figure 9.27 illustrate the practical step responses of the two digital control systems. This again confirms that the digital control system which possesses the highest figure of merit gives responses which are closest to the analogue equivalents.

9.6. Other digital control topologies

9.6.1. Digital control with the delta-operator

The delta operator [Fo91] has been introduced as a means of reducing coefficient sensitivity problems. The analysis which follows shows the effectiveness of the delta operator in control problems. The procedure discussed in Section 9.3, which is used to determine the best sampling rate, may be adapted to accommodate the delta operator.

9.6.2. Example 3

Consider the analogue control system of Figure 9.18. Using the delta operator and employing coefficients represented to just two significant bits, Figure 9.28 shows the
Improving the performance of digital controllers

error curves and the figure of merit for the control system for various sampling rates. The highest figure of merit occurs when \( T = 0.0240 \) seconds and a significantly poor figure of merit occurs for \( T = 0.0200 \) seconds. Incidentally, for both cases, the transfer function of the digital controller is

\[
C_q(\delta) = 4.0000 - 0.3750\delta^{-1}. \tag{9.24}
\]

Figures 9.29 and 9.30 respectively show the practical step responses of the digital control systems for \( T = 0.0240 \) seconds and \( T = 0.0200 \) seconds. With coefficients having just two significant bits, the digital controller with the highest figure of merit produces the best response and maintains a high fidelity with the continuous response.

9.6.3. Example 4.

The continuous compensator

\[
C(s) = \frac{0.30s + 1.80}{0.180s} \tag{9.25}
\]

gives adequate closed-loop step responses when cascaded with the plant

\[
P(s) = \frac{100}{s^2 + 20s + 100}. \tag{9.26}
\]

Using the error analysis techniques, a sampling window and hence the figure of merit for the digital controller can be readily determined; Figure 9.31 illustrates such results for coefficients having just two significant bits. From Figure 9.31, the sampling period \( T = 0.007 \) seconds produces a control system having a high figure of merit and the resulting digital controller is

\[
C_q(\delta) = 1.5000 + 0.0625\delta^{-1}. \tag{9.27}
\]

The figure of merit is significantly lower where \( T = 0.004 \) seconds and the digital controller is

\[
C_q(\delta) = 1.5000 + 0.03125\delta^{-1}. \tag{9.28}
\]
Improving the performance of digital controllers

Figure 9.33 shows the closed-loop step response for the digital control system which has a sampling interval of $T = 0.007$ seconds, and Figure 9.34 shows the equivalent response using $T = 0.004$ seconds. When compared with the closed-loop analogue response of Figure 9.32, the digital filter with the highest figure of merit again produces the closest response.

9.6.4 Two-Loop Control Strategies Using First-Derivative Feedback

Consider the control system of Figure 9.35 which models a two-loop control system with first-derivative feedback (e.g., velocity feedback in a position control system). The system contains an outer-loop controller $C_1(s)$ and an inner-loop controller $C_2(s)$. The inner loop compensator can be designed using classical control techniques (using the model of Figure 9.36) such that adequate first-derivative control may be achieved for the plant $P(s)$. Once this has been designed then the inner-loop can be modelled as a block $P_1(s)$ (see Figure 9.35) such that the outer-loop controller $C_2(s)$ may be designed using the usual techniques.

In order to design a digital control system to perform the equivalent function, the control system also needs to be developed in two stages. First, using w-plane design the inner-loop digital compensator is designed to give a first derivative control having the same phase margin and crossover frequency as the continuous system. Following this, a w-plane model $P_1(w)$ of the inner loop should be created.

The sampling interval of the outer control loop is $k$ times greater than that of the inner control loop (where $k$ is an integer and the product $kT$ satisfies the Nyquist sampling criterion). The outer-loop controller is designed by first translating $P_1(w)$ into a secondary w-plane corresponding to the sampling period of the outer loop. Using w-plane techniques the outer-loop digital controller is designed in the new w-plane such that the phase margin and cross-over frequency of the open-loop digital control system matches its analogue counterpart.
By using error analysis techniques on the overall control system, a sampling window may be established. For all values of T within the sampling window the figure of merit for the digital control system can be calculated as well as all permissible values of kT for the outer control loop. Thus the figure of merit is dependent on both the inner-loop sampling period T and k.

9.6.5. Example 5

Consider the continuous control system of Figure 9.37. Suppose a digital control system is to be designed using the z-operator and coefficients with 4-fractional bits then Figure 9.38 shows the error analysis required to determine the sampling window. Figure 9.39 shows a three dimensional plot illustrating how the figure of merit for the digital control system varies with parameters k and T. The ideal closed-loop step response of the analogue control system is given in Figure 9.40.

The figure of merit for the digital controller is highest, in this example, when the sampling period of the inner loop is $T = 0.0030$ seconds and the sampling period of the outer loop is 0.0300 seconds (i.e. $k = 10$). The resulting digital controllers are

$$C_1(z) = \frac{2.0000 - 1.9375z^{-1}}{1 - z^{-1}} \quad (9.29)$$

and

$$C_2(z) = \frac{1.5000 - 1.4375z^{-1}}{1 - z^{-1}} \quad (9.30)$$

Using the same value of T for the inner loop and setting $k = 1$, the outer digital controller has the transfer function

$$C_1(z) = \frac{1.9375 - 1.9375z^{-1}}{1 - z^{-1}} \quad (9.31)$$

Figure 9.40 shows how both digital control systems respond to a unit step input in addition to the continuous reference. The system with a highest figure of merit bears a close resemblance to the continuous reference. However, the system with the lower
Improving the performance of digital controllers

figure of merit gives a poor response, despite the outer loop sampling ten times faster. This emphasises the requirements of investigating both the inner and outer loop sampling rates in order to determine the system with the highest figure of merit.

9.6.6. Two loop control strategies controlling two physical systems

Consider the control system topology of Figure 9.41 which shows a two-loop control system with a compensated plant and compensated actuator. The inner loop ensures adequate actuator performance while the outer loop controller ensures that the overall control system behaves in an appropriate manner.

To design an analogue control system of such topology the inner loop compensator needs to be designed first. Once this has been designed the inner loop may be combined with the plant to form effectively a secondary plant $P_i(s)$. The outer loop compensator is designed for $P_i(s)$.

If a digital equivalent of this system is required then the inner loop digital controller is designed first using $w$-plane techniques such that the phase margin and crossover frequency for the digital compensator-actuator combination are the same as for the analogue equivalent. Assuming the sampling period of the inner loop is $T$ seconds and that of the outer loop is $kT$ seconds then the $w$-plane model of the inner loop and plant is formed for a sampling interval of $kT$ seconds. The outer loop digital controller is designed using $w$-plane techniques such that the phase margin and crossover frequency of the overall digital system match that of the continuous system.

To find the values of $k$ and $T$ which minimise the overall error, error analysis is performed on the inner loop to define the sampling window. Once the sampling window is defined the figure of merit may be calculated on the overall system to find the $(k, T)$ combination which yields the least error.

Consider the system of Figure 9.42 which shows a plant-actuator combination which is compensated by two PI compensators in a dual-loop configuration. Figure 9.43 shows the error analysis for the inner loop to define the sampling window using coefficients having four fractional bits. The variation of the figure of merit with \( k \) and \( T \) is shown in Figure 9.44. The highest figure of merit occurs when \( T = 0.005 \) seconds and \( k = 3 \). For this \((k, T)\) configuration, the inner loop digital controller is

\[
C_1(z) = \frac{4.6875 - 4.625z^{-1}}{1 - z^{-1}} \quad (9.32)
\]

and the outer loop digital controller is

\[
C_2(z) = \frac{4.9375 - 4.875z^{-1}}{1 - z^{-1}} \quad (9.33)
\]

For comparison, the equivalent outer loop digital compensator for \( T = 0.005 \) seconds and \( k = 1 \), giving an overall lower figure of merit, has the transfer function

\[
C_2(z) = \frac{4.875 - 4.875z^{-1}}{1 - z^{-1}} \quad (9.34)
\]

Figure 9.45 shows the closed-loop step response of the analogue control system along with the equivalent responses from the two digital control systems. The system with the highest figure of merit \((k = 3, T = 0.005)\) maintains a reasonably high fidelity with the analogue system even though the coefficients are specified to just four fractional bits. The system with \( k = 1 \) has a very poor response despite the fact that the sampling rate of the outer loop is three times faster than that for the system with the highest figure of merit.
Improving the performance of digital controllers

9.7. Summary

This Chapter has presented several methods by which the performance of digital controllers may be improved.

Various design methods were compared and the w-plane method of designing digital controllers is the most effective design technique out of those considered. This is because w-plane design takes into account the phase lag introduced by the zero-order hold.

A technique was presented to improve the performance of single-loop control systems. To determine the best range of sampling rates to be used in the digital controller, a sampling window needs to be determined where the algorithmic error and coefficient representation error are of similar orders of magnitude. For each value of $T$ inside the sampling window the figure of merit for the digital control system is determined to find the optimum sampling rate. Some wordlength considerations are discussed for the internal variable and quantiser requirements such that the control system behaves as required. The Chapter also presents the design of a software tool to automate this overall process.

The technique of improving the performance of digital controllers with respect to the analogue counterpart was extended to several other control strategies. For some of the examples in this Chapter, practical results were supplied to support the techniques of sampling period selection.
Improving the performance of digital controllers

Figure 9.1. The ideal continuous control system.

Figure 9.2. The digital control system.

Figure 9.3. Gain-phase plots illustrating the reduction of stability brought about by digitisation.
Improving the performance of digital controllers

Figure 9.4. Step responses illustrating the performance of digital design techniques.

Figure 9.5. Defining the deterministic errors in a digital control system.
Improving the performance of digital controllers

Figure 9.6. Variation of error magnitudes with sampling rate and illustration of the 'sampling window'.

Figure 9.7. Relating the optimum sampling window to the control system performance.
Improving the performance of digital controllers

Figure 9.8. Variation of the figure of merit with the sampling interval value for all values of T inside the sampling window.

Figure 9.9. Unit step responses for digital controllers realised using various sampling times and 8-bit coefficients.
Improving the performance of digital controllers

Figure 9.10. Top-level structure chart of the digital controller design hardware.

Figure 9.11. Structure chart for obtaining parameter data for the digital controller.
Improving the performance of digital controllers

Figure 9.12. Structure chart for the design of the digital controller.

Figure 9.13. Structure chart for selecting the required sampling range.

Figure 9.14. Structure chart for determining the figure of merit for the digital controllers.
Improving the performance of digital controllers

Figure 9.15. Structure chart for selecting the digital controller with the highest figure of merit.

Figure 9.16. Structure chart for selecting the ADC and DAC parameters.
Improving the performance of digital controllers

**Figure 9.17.** Structure chart for determining the internal variable requirements.

**Figure 9.18.** Practical Example 1.

**Figure 9.19.** Practical Example 2.
Improving the performance of digital controllers

Figure 9.20. The error curves, sampling window and figure of merit for the digital controller required for Example 1.

Figure 9.21. The error curves, sampling window and figure of merit for the digital controller required for Example 2.
Improving the performance of digital controllers

Figure 9.22. Unit step responses for the continuous control system and two digital control systems of Example 1.

Figure 9.23. Practical step responses for the system of Example 1 using $T = 0.0120$ seconds.

Figure 9.24. Practical step responses for the system of Example 1 using $T = 0.0130$ seconds.
Improving the performance of digital controllers

Figure 9.25. Unit step responses for the continuous control system and two digital control systems of Example 2.

Figure 9.26. Practical step responses for the system of Example 2 using $T = 0.0180$ seconds.

Figure 9.27. Practical step responses for the system of Example 2 using $T = 0.0150$ seconds.
Improving the performance of digital controllers

Figure 9.28. The error curves, sampling window and figure of merit for the control system of Figure 9.18 using the delta-operator.

Figure 9.29. Practical step responses for the system of Example 3 using $T = 0.0240$ seconds.

Figure 9.30. Practical step responses for the system of Example 3 using $T = 0.0200$ seconds.
Improving the performance of digital controllers

Figure 9.31. The error curves, sampling window and figure of merit for the control system of Example 4 using the delta-operator.

Figure 9.32. Closed-loop step response of the continuous control system of Example 4.

Figure 9.33. Example 4, $T = 0.007$ s.

Figure 9.34. Example 4, $T = 0.004$ s.
Improving the performance of digital controllers

Figure 9.35. Model of two-loop control system using first-derivative feedback.

Figure 9.36. Designing the inner loop controller.

Figure 9.37. Model of continuous control system for Example 5.
Improving the performance of digital controllers

Figure 9.38. Using error analysis to establish a sampling window for the system of Example 5.

Figure 9.39. Variation of figure of merit with sampling interval, T, and k for Example 5.
Improving the performance of digital controllers

Figure 9.40. Unit step responses of the continuous and discrete control systems of Example 5.

Figure 9.41. Block diagram of a two-loop control system employing plant and actuator.

Figure 9.42. Continuous control system for Example 6.
Figure 9.43. Using error analysis to establish a sampling window for the system of Example 6.

Figure 9.44. Variation of figure of merit with sampling interval, T, and k for Example 6.
Improving the performance of digital controllers

Figure 9.45. Unit step responses of the continuous and discrete control systems of Example 6.
The Thesis has reviewed and developed techniques for predicting values for each type of error generated in digital filters and digital controllers. With knowledge of how the errors vary over a wide range of sampling frequencies it is possible to locate a region of sampling frequencies where the errors are of similar orders of magnitude such that no particular error form dominates. Once that region has been established a figure of merit technique has been developed to select an appropriate value for the sampling interval which minimises the overall error generated by the resulting digital filter. This technique, which makes the best possible utilisation of the available hardware, has the potential to create digital filters with a high performance using modest hardware.

The TMS320C240 is a digital signal processor, designed by Texas Instruments, which is due for release in 1997 [TI96]. The TMS320C240 is a 16-bit fixed-point DSP which boasts two 10 bit ADC's which convert within 10μs, and 12 PWM outputs, making it an affordable single chip digital controller. If the techniques detailed in the Thesis, for choosing an appropriate value for the sampling period are applied, the TMS320C240 shows potential for very high performance digital control.
10.1 Original Contributions

Algorithmic error is an aspect of digital filter error analysis that has had very little coverage in the past. Chapter Two presented a detailed analysis of algorithmic error; the contents of which have recently been published [O195b].

Chapter Three introduced a quantity termed "Quantised-coefficient algorithmic error" which combines the two deterministic error forms, algorithmic error and coefficient representation error, into a more useful form.

An existing technique for estimating the average quantisation error in a digital filter was corrected and extended in Chapter Four. In addition to being simple to apply, examples have shown this method of computing ADC quantisation error and multiple-word truncation error to be more accurate than other methods.

There is currently disagreement about which form of the delta operator is superior. Chapter Five made a comparative study between the two definitions to resolve the argument and some of the work contained in the chapter has already been published [O196].

Chapter Seven introduced a novel spectral technique for separating the deterministic error (algorithmic error and coefficient representation error) from the stochastic error (ADC quantisation error) in a digital filter. The development of normalised DFT and inverse NDFT techniques facilitated this process. This spectral technique is useful for validating the theoretical error analysis undertaken on a practical digital filter. However, it has been shown that the sensitivity to small changes in signal frequency and amplitude is critical and requires further investigation.

Chapter Eight presented a means of selecting a value for the sampling period of a digital filter using a "Figure of merit" to minimise the overall error. Even though some of this work has already been published [O195a], the thesis presents two improved
definitions for the figure of merit. The Chapter also revealed anomalies in that excessively long coefficient wordlengths could impair the performance of a digital filter.

Chapter Nine extended the figure of merit concept, covered in the previous chapter, to improve the performance of closed-loop digital control systems with respect to the analogue equivalent. By carefully selecting a suitable value for the sampling rate, the performance of the digital control system can be optimised for given hardware.

10.2. Recommendations for further work

The ideas in the thesis relate to systems presented in transfer function format. The natural progression is to extend the ideas to state-space. For the single-input-single-output (SISO) case, the state-space formulation can be translated into a transfer function and the error analysis would be straightforward.

The extension of state-space error analysis could then be applied to multiple-input-multiple-output (MIMO) systems. However, it is anticipated that the transition from SISO to MIMO will be great and care should be exercised to ensure the MIMO system is stable.

High performance dynamics is an important feature of certain commercial control schemes such as engine management systems and the cost of such systems can be reduced significantly if less powerful processors are used. The use of the figure of merit technique could lead to the choice of a less expensive processor and great savings in large volume production. Future work would involve applying the ideas of the thesis to mass produced commercial control systems to simultaneously minimise cost and maximise performance.
REFERENCES


Fer80a  Ferranti Semiconductors, ZN426 data sheet, 1980.

Fer80b  Ferranti Semiconductors, ZN428 data sheet, 1980.


References


Kim94a Kim, S. and Sung, W., "A floating-point to fixed-point assembly


References


A.1. Introduction

In the stochastic computation of quantisation noise propagation through a digital filter, one of the methods that can be used relates to the autocorrelation function between the point where noise is introduced to the output. To calculate the relevant autocorrelation function it is advantageous to know the results of some infinite summations. This appendix shows the calculations to illustrate how the relevant infinite series are determined.

The technique of determining the result of an infinite series, highlighted in [Gr70] relies on the convergence of the series. Convergence is taken for granted in stable systems as the poles lie inside the unit circle.
A.2. Infinite series (i)

Consider the infinite summation

\[ \sum_{k=0}^{\infty} x^k. \]

In order to compute the result of this summation, firstly consider the summation of the first N terms. Let

\[ S_N = \sum_{k=0}^{N} x^k = 1 + x + x^2 + \ldots + x^N. \] \hspace{1cm} (A.1)

Multiplying this expression by x gives

\[ xS_N = x + x^2 + x^3 + \ldots + x^N + x^{N+1}. \] \hspace{1cm} (A.2)

Subtracting (A.2) from (A.1) gives

\[ S_N(1 - x) = 1 - x^{N+1}. \] \hspace{1cm} (A.3)

Now as N tends towards infinity

\[ S_\infty = \sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}. \] \hspace{1cm} (A.4)

A.3. Infinite series (ii)

Consider the infinite summation

\[ \sum_{k=0}^{\infty} kx^{k-1} \]

Consider the summation of the first N terms. Letting

\[ S_N = \sum_{k=0}^{N} kx^{k-1} = 0 + 1 + 2x + 3x^2 + \ldots + Nx^{N-1} \] \hspace{1cm} (A.5)

and multiplying this by x gives
Appendix A: Infinite series

\[ xS_N = x + 2x^2 + \ldots + (N - 1)x^{N-1} + Nx^N. \]  \hfill (A.6)

Subtracting (A.6) from (A.5) results in

\[ S_N(1 - x) = (1 - Nx^N) + (x + x^2 + x^3 + \ldots + x^N). \]  \hfill (A.7)

As \( N \) tends towards infinity equation (A.7) becomes

\[ S_N(1 - x) = 1 + x + x^2 + x^3 + \ldots \]
\[ = \frac{1}{1 - x}. \]  \hfill (A.8)

Therefore

\[ S_N = \sum_{k=0}^{\infty} kx^{k-1} = \frac{1}{(1 - x)^2}. \]  \hfill (A.9)

A.4. Infinite series (iii)

Consider the infinite series

\[ \sum_{k=0}^{\infty} kx^k. \]

Using equation (A.9),

\[ \sum_{k=0}^{\infty} kx^k = x\sum_{k=0}^{\infty} kx^{k-1} = \frac{x}{(1 - x)^2}. \]  \hfill (A.10)
A.5. Infinite series (iv)

Consider the infinite series

\[
\sum_{k=0}^{\infty} k^2 x^{k-1}
\]

By examining the summation of the first \( N \) terms, let

\[
S_N = \sum_{k=0}^{N} k^2 x^{k-1} = 0 + 1 + 4x + 9x^2 + 16x^3 + \ldots + N^2 x^{N-1}. \tag{A.11}
\]

Multiplying by \( x \) gives

\[
xS_N = x + 4x^2 + 9x^3 + \ldots + (N - 1)^2 x^{N-1} + N^2 x^N. \tag{A.12}
\]

Subtracting (A.12) from (A.11) results in

\[
S_N(1 - x) = (1 - N^2 x^N) + 3x + 5x^2 + 7x^3 + \ldots + (2N - 1)x^{N-1}
= \sum_{k=1}^{N} (2k - 1)x^{k-1} - N^2 x^N. \tag{A.13}
\]

As \( N \) tends towards infinity, using the results of equations (A.4) and (A.9), equation (A.14) becomes

\[
S_\infty(1 - x) = 2\sum_{k=1}^{\infty} kx^{k-1} - \sum_{k=0}^{\infty} x^k
= 2\left( \frac{1}{(1 - x)^2} \right) - \frac{1}{1 - x}
= \frac{1 + x}{(1 - x)^3}. \tag{A.14}
\]

Therefore

\[
\sum_{k=0}^{\infty} k^2 x^{k-1} = \frac{1 + x}{(1 - x)^3}. \tag{A.15}
\]
Appendix A: Infinite series

A.6 Infinite series (v)

Consider the infinite summation

\[ \sum_{k=0}^{\infty} k^3 x^{k-1} \]

Let \( S_N \) represent the first \( N \) terms of the summation then

\[
S_N = 0 + 1 + 8x + 27x^2 + 64x^3 + \ldots + N^3 x^{N-1}. \tag{A.16}
\]

Multiplying each term in equation (A.16) by \( x \) gives

\[
xS_N = x + 8x^2 + 27x^3 + \ldots + (N - 1)^3 x^{N-1} + N^3 x^N. \tag{A.17}
\]

Subtracting equation (A.17) from (A.16) results in equation (A.18) below

\[
S_N(1 - x) = (1 - N^3 x^N) + 7x + 19x^2 + 37x^3 + \ldots + (3N^2 - 3N + 1)x^{N-1}
\]

\[
= \sum_{k=1}^{N} (3k^2 - 3k + 1)x^{k-1} - N^3 x^N
\]

As the number of terms approaches infinity and applying the results of equations (A.4), (A.9) and (A.15), equation (A.18) becomes

\[
S_\infty(1 - x) = 3\sum_{k=0}^{\infty} k^2 x^{k-1} - 3\sum_{k=0}^{\infty} kx^{k-1} + \sum_{k=0}^{\infty} x^k
\]

\[
= 3\frac{(1 + x)}{(1 - x)^3} - 3\frac{x}{(1 - x)^2} + \frac{1}{(1 - x)} \tag{A.19}
\]

\[
= \frac{x^2 + 4x + 1}{(1 - x)^3}.
\]

Therefore

\[
S_\infty = \sum_{k=0}^{\infty} k^3 x^{N-1} = \frac{x^2 + 4x + 1}{(1 - x)^4}. \tag{A.20}
\]
A.7. Infinite series (vi)

Consider the infinite series

\[ \sum_{k=1}^{\infty} \left( \sum_{i=1}^{k} i \right)^2 x^{k-1}. \]

Letting \( S_N \) be the summation of the first \( N \) terms in the series gives

\[ S_N = 1 + 9x + 36x^2 + 100x^3 + \ldots + \left( \sum_{i=1}^{N} i \right)^2 x^{N-1}. \]  (A.21)

Multiplying through by \( x \) gives

\[ xS_N = x + 9x^2 + 36x^3 + \ldots + \left( \sum_{i=1}^{N-1} i \right)^2 x^{N-1} + \left( \sum_{i=1}^{N} i \right)^2 x^N. \]  (A.22)

Subtracting (A.22) from (A.21) results in

\[ S_N(1-x) = \left( 1 - \left( \sum_{i=1}^{N} i \right)^2 x^N \right) + 8x + 27x^2 + 64x^3 + \ldots + N^3x^{N-1} \]

\[ = \sum_{k=1}^{N} k^3x^{k-1} - \sum_{i=1}^{N} i^2 x^N. \]  (A.22)

Now as \( N \) approaches infinity and using the result of equation (A.20)

\[ S_{\infty}(1-x) = \sum_{k=1}^{\infty} k^3x^{k-1} = \frac{x^2 + 4x + 1}{(1-x)^4}. \]  (A.23)

Therefore

\[ S_{\infty} = \sum_{k=1}^{\infty} \left( \sum_{i=1}^{k} i \right)^2 x^{k-1} = \frac{x^2 + 4x + 1}{(1-x)^5}. \]  (A.24)
A.8. Infinite series (vii)

Consider the infinite series

\[ \sum_{k=1}^{\infty} \sum_{i=1}^{k} ix^{k-1} \]

To evaluate this summation, as before, the first \( N \) terms of the summation will be considered. That is

\[ S_N = 1 + 3x + 6x^2 + 10x^3 + 15x^4 + ... + \sum_{i=1}^{N} ix^{N-1}. \quad (A.25) \]

Multiplying each term in (A.25) gives

\[ xS_N = x + 3x^2 + 6x^3 + 10x^4 + ... + \sum_{i=1}^{N-1} ix^{N-1} + \sum_{i=1}^{N} ix^{N}. \quad (A.26) \]

Subtracting (A.26) from (A.25) gives the equation (A.27) below

\[ S_N(1 - x) = \left( 1 - \sum_{i=1}^{N} ix^{N} \right) + 2x + 3x^2 + 4x^3 + 5x^4 + ... + Nx^{N-1}. \]

As \( N \) approaches infinity

\[ S_\infty = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + ... \]

\[ \quad = \sum_{k=1}^{\infty} kx^{k-1} \]

\[ \quad = \frac{1}{(1 - x)^2}. \quad (A.27) \]

after application of equation (A.9). Therefore

\[ S_\infty = \sum_{k=1}^{\infty} \sum_{i=1}^{k} ix^{k-1} = \frac{1}{(1 - x)^3}. \quad (A.28) \]
A.9. Infinite series (viii)

Consider the infinite series

\[ \sum_{k=1}^{\infty} \sum_{i=1}^{k-1} ix^{k-1} \]

The sum of the first \( N \) terms of this series, \( S_N \), is

\[ S_N = 0 + x + 3x^2 + 6x^3 + 10x^4 + 15x^5 + \ldots + \sum_{i=1}^{N-1} ix^{N-1}. \]  

(A.29)

On comparison with equation (A.25), and letting \( N \to \infty \)

\[ S_\infty = x \sum_{k=1}^{\infty} \sum_{i=1}^{k-1} ix^{k-1}. \]  

(A.30)

From equations (A.30) and (A.28)

\[ S_\infty = \sum_{k=1}^{\infty} \sum_{i=1}^{k-1} ix^{k} = \frac{x}{(1 - x)^3}. \]  

(A.31)

A.1.10 Infinite series (ix)

Finally, consider the infinite series

\[ \sum_{k=1}^{\infty} k \sum_{i=1}^{k-1} ix^{k-1} \]

Again, let \( S_N \) represent the first \( N \) terms of this infinite series, where

\[ S_N = 1 + 6x + 18x^2 + 40x^3 + 75x^4 + \ldots + N \sum_{i=1}^{N} ix^{N-1}. \]  

(A.32)

Multiplying expression (A.32) through by \( x \) results in
\[ xS_N = x + 6x^2 + 18x^3 + 40x^4 + \ldots + (N-1)\sum_{i=1}^{N-1} ix^{N-1} + N\sum_{i=1}^{N} ix^{N}. \quad (A.33) \]

Subtracting equation (A.33) from equation (A.32) gives

\[
S_N(1-x) = \left( 1 - N\sum_{i=1}^{N} ix^{N} \right) + 5x + 12x^2 + 22x^3 + \ldots + \left( N\sum_{i=1}^{N} i - (N-1)\sum_{i=1}^{N-1} i \right)x^{N-1}.
\]

But

\[
N\sum_{i=1}^{N} ix^{N-1} - (N-1)\sum_{i=1}^{N-1} ix^{N-1} = \left( N\sum_{i=1}^{N} i - N\sum_{i=1}^{N-1} i \right) + \sum_{i=1}^{N-1} i x^{N-1}
\]

\[
= \left( N^2 + \sum_{i=1}^{N-1} i \right)x^{N-1}.
\]

Letting \( N \to \infty \), and using the results of equations (A.35), (A.15) and (A.31), equation (A.34) becomes

\[
S_N(1-x) = 1 + 5x + 12x^2 + \ldots + \left( N^2 + \sum_{i=1}^{N-1} i \right)x^{N-1} + \ldots
\]

\[
= \sum_{k=1}^{\infty} k^2x^{k-1} + \sum_{k=1}^{\infty} k\sum_{i=1}^{k} ix^{k-1}
\]

\[
= \frac{1 + x}{(1-x)^3} + \frac{x}{(1+x)^3}.
\]

Therefore

\[
S_N = \sum_{k=1}^{\infty} k\sum_{i=1}^{k} ix^{k-1} = \frac{2x + 1}{(1-x)^4}.
\]

\[ \text{Appendix A: Infinite series} \]

\[ \text{238} \]
B.1 Introduction

Chapter Four presented the corrected technique to numerically evaluate the contour integral

$$\Phi_0 = \frac{1}{2\pi j} \oint_{|z|=1} H(z)H(z^{-1})z^{-1}dz$$  \hspace{1cm} (B.1)

for systems with simple poles. The results for systems with multiple poles were stated in the Chapter but were not derived.

This Appendix derives the matrices required for evaluating the contour integral of equation (B.1) for second- and third-order systems with double poles and a third-order system with a triple pole.
Appendix B: Quantisation Noise Analysis For Systems With Multiple Poles

B.2. Second-order systems with double poles

Consider a second-order system which, when expanded into partial fractions, is

\[ H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{r_2}{(z - p_1)^2} \]  

\[ = H_0(z) + H_1(z) + H_2(z). \]  

(B.2)

Examining the impulse responses of the three terms in equation (B.2) gives

\[ h_0(k) = \{ r_0, 0, 0, 0, \ldots \} \]  

\[ h_1(k) = \{0, r_1, r_1p_1, r_1p_1^2, r_1p_1^3, \ldots \} \]  

\[ h_2(k) = \{0, 0, r_2, 2r_2p_1, 3r_2p_1^2, 4r_2p_1^3, \ldots \}. \]  

(B.3)

The \( k^{th} \) sample of each of these impulse responses is therefore

\[ h_0(k) = r_0 0^k \]  

\[ h_1(k) = r_1p_1^{k-1} \]  

\[ h_2(k) = (k-1)r_2p_1^{k-2}. \]  

(B.4)  

(B.5)  

(B.6)

But

\[ p_1^{k-1} \geq 0 \quad \forall k < 1 \]  

\[ p_1^{k-2} \geq 0 \quad \forall k < 2. \]  

(B.7)

Now, the \( k^{th} \) sample of the square of the impulse response is
Appendix B: Quantisation Noise Analysis For Systems With Multiple Poles

\[
[h(k)]^2 = \left[r_0 0^k + r_1 p_1^{k-1} + (k-1)r_2 p_1^{k-2}\right] \left[r_0 0^k + r_1 p_1^{k-1} + (k-1)r_2 p_1^{k-2}\right] \\
= r_0^2 (0)^k + r_1^2 p_1^{k-1} p_1^{k-1} + 2(k-1)r_1 r_2 p_1^{k-1} p_1^{k-2} + (k-1)^2 r_2^2 p_1^{k-2} p_1^{k-2} \\
= r_0^2 (0)^k + r_1^2 [p_1^{k-1}]^2 + 2(k-1)r_1 r_2 [p_1^{k-2}]^2 + (k-1)^2 r_2^2 [p_1^{k-2}]^2 
\]  
(B.8)

It follows that

\[
\Phi_0 = \sum_{k=0}^{\infty} [h(k)]^2 = [R] [Q] [R]^T \\
= r_0^2 + \frac{r_1^2}{1 - p_1^2} + \frac{2r_1 r_2}{(1 - p_1^2)^2} + r_2^2 \left\{ \frac{1 + p_1^2}{(1 - p_1^2)} \right\} 
\]  
(B.9)

using the infinite series of Appendix A. The [R] and [Q] matrices are respectively given by

\[
[R] = \begin{bmatrix} r_0 & r_1 & r_2 \end{bmatrix} 
\]  
(B.10)

and

\[
[Q] = \begin{bmatrix} 1 & 0 & 0 \\
0 & \frac{1}{1 - p_1^2} & \frac{p_1}{(1 - p_1^2)^2} \\
0 & \frac{p_1}{(1 - p_1^2)^2} & \frac{1 + p_1^2}{(1 - p_1^2)^3} \end{bmatrix} 
\]  
(B.11)
B.3. Third-order systems with double pole

Consider the general third-order transfer function having a double pole

$$H(z) = r_0 + \frac{r_1}{z - p_1} + \frac{r_2}{z - p_2} + \frac{r_3}{(z - p_2)^2}$$

$$= h_0(z) + h_1(z) + h_2(z) + h_3(z)$$

$$= h(z).$$

After simplification, the $k^{th}$ sample of the impulse response squared is

$$[h(k)]^2 = [h_0(k)]^2 + [h_1(k)]^2 + [h_2(k)]^2 + [h_3(k)]^2$$

$$+ 2 [h_1(k)h_2(k) + h_2(k)h_3(k) + h_1(k)h_3(k)]$$

Taking each of the terms in equation (B.13) and summing each over the interval $k=0$ to $k = \infty$, using Appendix A gives

$$\sum_{k=0}^{\infty} [h_0(k)]^2 = r_0^2$$

$$\sum_{k=0}^{\infty} [h_1(k)]^2 = r_1^2 \left( \frac{1}{1 - p_1^2} \right)$$

$$\sum_{k=0}^{\infty} [h_2(k)]^2 = r_2^2 \left( \frac{1}{1 - p_2^2} \right)$$

$$\sum_{k=0}^{\infty} [h_3(k)]^2 = r_3^2 \left( \frac{1 + p_2^2}{(1 - p_2^2)^3} \right)$$

$$\sum_{k=0}^{\infty} [h_1(k)h_2(k)] = r_1r_2 \left( \frac{1}{1 - p_1p_2} \right)$$
Appendix B: Quantisation Noise Analysis For Systems With Multiple Poles

\[ \sum_{k=0}^{\infty} [h_2(k)h_3(k)] = r_2r_3 \left( \frac{p_2}{1 - p_2^2} \right) \]  \hspace{1cm} (B.19)

\[ \sum_{k=0}^{\infty} [h_1(k)h_3(k)] = \sum_{k=0}^{\infty} [r_2p_1^{k-1} \cdot r_3(k-1)p_2^{k-2}] \]

\[ = r_1r_3 \left( \frac{p_1}{1 - p_1p_2} \right) \]  \hspace{1cm} (B.20)

The contour integral of equation (B.1) can now be evaluated as

\[ \Phi_0 = \sum_{k=0}^{\infty} [h(k)]^2 = [R] \ [Q] \ [R]^T \]  \hspace{1cm} (B.21)

where

\[ R = [r_0 \ r_1 \ r_2 \ r_3] \]  \hspace{1cm} (B.22)

and

\[ Q = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \frac{1}{1 - p_1^2} & \frac{1}{1 - p_1p_2} & \frac{p_1}{(1 - p_1p_2)^2} \\
0 & \frac{1}{1 - p_1p_2} & \frac{1}{1 - p_2^2} & \frac{p_2}{(1 - p_2^2)^2} \\
0 & \frac{p_1}{(1 - p_1p_2)^2} & \frac{p_2}{(1 - p_2^2)^2} & \frac{1 + p_2^2}{(1 - p_2^2)^3}
\end{bmatrix} \]  \hspace{1cm} (B.23)
B.4. Third-order systems with triple pole

Consider, finally a general third-order digital transfer function with a single pole located at the point $z = p$:

$$H(z) = r_0 + \frac{r_1}{z - p} + \frac{r_2}{(z - p)^2} + \frac{r_3}{(z - p)^3}$$

$$= h_0(z) + h_1(z) + h_2(z) + h_3(z)$$

$$= h(z).$$

Examination of the $h_3(z)$ term reveals the impulse response

$$h_3(k) = r_3 \{0, 0, 1, 3p, 6p^2, 10p^3, ... \}$$

$$= r_3 p^{k-3} \sum_{i=1}^{k-2} i$$

However, for this analysis

$$p^{k-3} \triangleq 0 \quad \forall \ k < 3.$$ (B.26)

Examining the square of the $k^{th}$ sample of the impulse response gives after some simplification

$$[h(k)]^2 = [h_0(k) + h_1(k) + h_2(k) + h_3(k)]^2$$

$$= [h_0(k)]^2 + [h_1(k)]^2 + [h_2(k)]^2 + [h_3(k)]^2$$

$$+ 2 [h_1(k)h_2(k) + h_1(k)h_3(k) + h_2(k)h_3(k)]$$

(B.27)

From earlier analysis it can readily be shown that

$$\sum_{k=0}^{\infty} [h_0(k)]^2 = r_0^2.$$ (B.28)
Appendix B: Quantisation Noise Analysis For Systems With Multiple Poles

\[ \sum_{k=0}^{\infty} |h_1(k)|^2 = r_1^2 \left\{ \frac{1}{1 - p^2} \right\}. \quad (B.29) \]

\[ \sum_{k=0}^{\infty} |h_2(k)|^2 = r_2^2 \left\{ \frac{1 + p^2}{(1 - p^2)^3} \right\}. \quad (B.30) \]

\[ \sum_{k=0}^{\infty} [h_1(k)h_2(k)] = r_1r_2 \left\{ \frac{p}{(1 - p^2)^2} \right\}. \quad (B.31) \]

The rest of the terms are calculated as

\[ \sum_{k=0}^{\infty} |h_3(k)|^2 = \sum_{k=0}^{\infty} r_3^2 p^{-3} \sum_{i=1}^{k-2} i^2 \]
\[ = r_3^2 \left\{ \frac{1 + 4p^2 + p^4}{(1 - p^2)^5} \right\}. \quad (B.32) \]

\[ \sum_{k=0}^{\infty} [h_2(k)h(3)] = r_2r_3 \sum_{k=1}^{\infty} (k-1)p^{k-3}p^{k-2} \sum_{i=1}^{k-2} i \]
\[ = r_2r_3p \sum_{k=1}^{\infty} (k-1)p^{2k-3} \sum_{i=1}^{k-2} i \]
\[ = r_2r_3 \left\{ \frac{2p^3 + p}{(1 - p^2)^4} \right\}. \quad (B.33) \]

The value of the contour integral \([R][Q][R]^T\) may be found in the usual way with

and

245
Appendix B: Quantisation Noise Analysis For Systems With Multiple Poles

\[
\sum_{k=0}^{\infty} [h_1(k)h_3(k)] = r_1r_3 \sum_{k=1}^{\infty} p^{k-1} p^{k-3} \sum_{i=1}^{k-2} i
\]

\[
= r_1r_3 p^2 \sum_{k=1}^{\infty} \sum_{i=1}^{k-2} i [p^2]^{k-3}
\]

\[
= r_1r_3 \left[ \frac{p^2}{(1 - p^2)^3} \right]
\]

\[
R = [r_0, r_1, r_2, r_3]
\]  

\[
[Q] = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \frac{1}{1 - p^2} & \frac{p}{(1 - p^2)^2} & \frac{p^2}{(1 - p^2)^3} \\
0 & \frac{p}{(1 - p^2)^2} & \frac{1 + p^2}{(1 - p^2)^3} & \frac{2p^3 + p}{(1 - p^2)^4} \\
0 & \frac{p^2}{(1 - p^2)^3} & \frac{2p^3 + p}{(1 - p^2)^4} & \frac{1 + 4p^2 + p^4}{(1 - p^2)^5}
\end{bmatrix}
\]

Clearly the same process could be applied to transfer functions of higher order with considerations of further infinite series. However they will not be considered in this thesis as higher order filters are considered as cascaded or parallel configurations of lower-order sections.
This appendix presents the full schematic diagrams of the circuits described in Chapter Six.

Figure C1 presents the circuit diagram for the timing module for the digital controller.

Figure C2 is the schematic for the analogue-to-digital converter, the digital-to-analogue converter and the associated control logic.

Figure C3 shows the schematic for an experimental control system. This is extended to include basic circuitry that can apply a unit step input.

The remaining diagrams illustrate circuitry to model second-order systems. Figure C4 is a circuit diagram for a second-order lightly-damped system, whilst Figure C5 illustrates the schematic for a second-order Type I system.
Figure C1. Circuit diagram of the timer for the digital controller
Appendix C: Circuit Diagrams

Figure C2. Circuit Diagram of the analogue-to-digital converter, the digital-to-analogue converter and the associated control logic.
Figure C3. Circuit diagram of an experimental control system model.
Figure C4. Circuit diagram for a second-order lightly-damped system.

Figure C5. Circuit diagram for a second-order, Type I system.
D.1. Introduction

This appendix discusses a useful piece of software written in MATLAB which can be used to design simple continuous compensators. The controller is designed with frequency-domain techniques using gain-phase plots.

The procedure to design compensators using this software is fairly simple. First, the user needs to enter the numerator and denominator coefficients of the plant transfer function. The gain-phase plot of the system is then displayed on the screen. The user can then zoom in and examine a particular region of the plot. This is done by 'clicking' and marking the top-left and bottom-right corners of the area of interest. The user needs to 'click' on a point on the (uncompensated) frequency response, denoted 'R' which will be the frequency point that the controller will be designed about. The user then clicks on a desired point (denoted 'P') through which the compensated frequency response should pass. The compensator is designed using magnitude and phase shifts required to carry the open-loop frequency response from R to P for the frequency value at point R.
D.2. Software Design

Figure D.1 illustrates the top-level structure chart for the software to design analogue controllers.

On execution, the routine asks the user to enter the numerator and denominator coefficients for the plant model. These are stored as variables NUM and DEN respectively. The gain versus phase plot of the system is then drawn on the screen and the results of the frequency response are stored as vectors of frequency, phase and magnitude.

The software then asks the user to select a point on the uncompensated frequency response. There are three main stages to this, as shown in Figure D.2. The first stage is to zoom in on the plot. The user is prompted to mark (by clicking) the top-left and top-right points on the plot to denote the region which requires enlarging. Using the 'ginput' command in MATLAB, the x- and y- coordinates can be readily found. These coordinates are then used to rescale the required region of the graph using the 'axis' command. The second stage requires the user to click on a point on the frequency response. Again using the 'ginput' command, the (x, y) coordinates of that point are easily found. The final stage is to find the frequency of the point along with the magnitude and the phase. By using least-squares the values of phase and magnitude \((\phi_R, M_R)\) closest to the values \((x, y)\) reveals the value of the frequency \(\omega_0\).

The software will then ask the user to click on the point where the compensated response should pass through. The \((x, y)\) coordinate of that point gives the phase and magnitude values of \((\phi_p, M_p)\) respectively.

Figure D.3 is a structure chart which shows how the transfer function of the compensator is determined. Firstly the phase difference is calculated using
The magnitude difference is calculated using
\[ M_{db} = M_p - M_R \]  \hspace{1cm} (D.2)
and is converted from decibels to a linear quantity, M.

The required number of cascaded compensators is then determined to ensure that the phase shift performed by each individual compensator does not exceed sixty degrees. If the required phase shift exceeds sixty degrees then compensator will be split into two or more identical first-order sections to perform the phase shift.

If the phase difference is positive a phase advance compensator is designed. A continuous phase advance compensator has the transfer function
\[ C(s) = G \frac{1 + skT_c}{1 + sT_c} \]  \hspace{1cm} (D.3)
where
\[ k = \frac{1 + \sin\phi}{1 - \sin\phi} \]  \hspace{1cm} (D.4)
\[ T_c = \frac{1}{\omega_c\sqrt{k}} \]  \hspace{1cm} (D.5)
and
\[ G = \frac{M}{\sqrt{k}}. \]  \hspace{1cm} (D.6)

For a negative phase difference, a proportional plus integral (P+I) compensator is designed with general transfer function
Appendix D: Analogue compensator design software

\[ C(s) = G \left[ \frac{1 + sT_c}{sT_c} \right] \]  \hspace{1cm} (D.7)

where

\[ \theta = 90^\circ + \phi \]  \hspace{1cm} (D.8)

\[ T_c = \frac{\tan \theta}{\omega_c} \]  \hspace{1cm} (D.9)

and

\[ G = M \sin \theta. \]  \hspace{1cm} (D.10)

If the phase difference is zero, a simple proportional controller needs designing with gain equal to M.

The complete analogue compensator is completed by cascading individual first-order units if the required phase shift exceeds sixty degrees. The value of the transfer function is then displayed on the screen.

Data concerning the designed system is then calculated and sent to the screen, as depicted in Figure D.4. Firstly the open-loop transfer function of the cascaded compensator and plant is calculated and displayed. The gain-phase plot of this compensated system is plotted on the same set of axes as the uncompensated system. The closed-loop transfer function of the compensated system is then calculated and displayed on the screen.

Finally, time-domain responses of the compensated system are carried out to determine the success of the compensator design. The closed-loop transfer function is calculated as a measure to compare the effectiveness of the compensated system. Unit step and unit ramp simulations are finally carried out on the compensated and uncompensated systems.
Figure D.9 gives the MATLAB code listing of this program.

### D.3. Example

Consider a plant with transfer function

\[
P(s) = \frac{3}{s^2 + 4s + 3}.
\]

(D.11)

Using the Analogue Control Design software, the gain-phase plots of the compensated and uncompensated system are shown in Figure D.6. Point R is the point on the uncompensated plot that was selected, point P is the desired point through which the compensated system should pass. The software designed a P+I compensator with transfer function

\[
C(s) = \frac{2.5383s + 3.0118}{0.8428s}.
\]

(D.12)

Figure D.7 shows the unit-step responses for the uncompensated and compensated systems. The compensated response meets the target response of unity within a reasonable time with about 21% overshoot. The uncompensated system only achieves 50% of the target response.

Figure D.8 shows the unit-ramp responses for the uncompensated and compensated systems. Neither system tracks the ramp successfully. The error between the ideal ramp and the uncompensated response increases with time while the error between the ideal ramp and the compensated response settles to a smaller fixed-value.

### D.4. Summary

This appendix has provided the software design for a program to design analogue compensators. The source code can be found at the back of this appendix.
Appendix D: Analogue compensator design software

Figure D.1. Top-level structure chart showing the overall structure of the analogue controller design software

Figure D.2. Structure chart of software to acquire a frequency point on the uncompensated response
Appendix D: Analogue compensator design software

Figure D.3. Structure chart of software for calculating the compensator transfer function.

Figure D.4. Structure chart for the software for displaying data about the control system.
Appendix D: Analogue compensator design software

Figure D.5. Structure chart for the software to display the time responses of the compensated and uncompensated systems.

Figure D.6. Gain versus phase plots for uncompensated and compensated systems (open-loop) produced by the Analogue Control Design software.
Figure D.7. Unit step responses for uncompensated and compensated closed-loop systems using the Analogue Control Design software.

Figure D.8. Unit ramp responses for uncompensated and compensated closed-loop systems using the Analogue Control Design software.
Figure D.9. MATLAB source code for the analogue controller design tool

```matlab
% ***********************************************
% * Analogue Controller Design *
% * By Mike A. Oliver *
% * 10th May 1996 *
% ***********************************************

clear;

num=input('Enter numerator of plant Transfer Function: ');
den=input('Enter denominator of plant Transfer Function: ');
figure(1);
clf;

class;

% ***********************************************
% * Asks user to enter the numerator and denominator *
% * of the plant transfer function *
% ***********************************************

class;

% ***********************************************
% * Plot the Nichols Chart of the uncompensated system *
% ***********************************************

Mag,Phase,w=nichols(num,den);
figure(1);
class;

Mag=20*log10(Mag);
hold on;
AX=axis;
plot([-180 180],AX(3:4),'--w');
plot([-180 180],AX(3:4),'--w');
title('Nichols Chart of uncompensated system');

class;

% ***********************************************
% * Zoom in on the plot *
% ***********************************************

class;

% ***********************************************
% * Prompt the user to select point with mouse *
% ***********************************************

class;

% ***********************************************
% * Getting a point with the mouse and finding the nearest *
% * value for the angular frequency using least-squares *
% * technique. *
% ***********************************************

[p,m]=ginput(1);
[temp,j]=min((Mag-m).^2+(Phase-p).^2);
MagR=Mag(j);
PhaseR=Phase(j);
text(PhaseR+10,MagR,'R');
plot(PhaseR,MagR,'o');
wc=w(j);
disp('Chosen Frequency: ');
disp(wc);
disp(' ');
```

261
Appendix D: Analogue compensator design software

% **************************************************************
% * Obtaining a point on the Nichols chart using the mouse for *
% * the compensated plot to pass through - denoted point P. *
% **************************************************************
disp(' '); disp('Use the mouse to select a point ' 'P' for the compensated response'); disp(''); [PhaseP,MagP]=ginput(1); text(PhaseP+10,MagP,'P'); plot(PhaseP,MagP,'om');

PhaseDifference=PhaseP-PhaseR;
MagDifference=MagP-MagR;

disp(' '); disp('Plant Transfer Function'); printsys(nurn,den,'s'); disp('');

% **************************************************************
% * Determining the number of cascaded compensators *
% * required *
% **************************************************************
comp=floor(abs(PhaseDifference)/60)+1;
PhaseDifference=PhaseDifference/comp;
MagDifference=MagDifference/comp;
M=10^4*MagDifference/comp;

% **************************************************************
% * If phase difference is negative, design a P+I compensator *
% **************************************************************
if PhaseDifference<0
    theta=90+PhaseDifference;
    theta=theta*pi/180;
    Tc=tan(theta)/wc;
    K=M*sin(theta);
    disp('P+I compensator designed. Transfer function:- ');
    numc=[1];
    denc=[1];
    for i=1:comp
        numc=conv(numc,K*[Tc 1]);
        denc=conv(denc,[Tc 1]);
    end;
    disp('Compensator');
    printsys(numc,denc,'s');
end

% **************************************************************
% * If phase difference is positive, design a phase advance *
% * compensator *
% **************************************************************
if PhaseDifference>0
    theta=PhaseDifference*3.1415926536(180,
    k=(1+sin(theta))/(1-sin(theta));
    Tc=1/(wc*sqrt(k));
    G=M/sqrt(k);
    disp('Phase advance compensator designed. Transfer function:- ');
    numc=[1];
    denc=[1];
    for i=1:comp
        numc=conv(numc,G*[Tc k 1]);
        denc=conv(denc,[Tc 1]);
    end;
    disp('Compensator');
    printsys(numc,denc,'s');
end

% **************************************************************
% * If there is no phase difference, design a proportional controller *
% **************************************************************
if PhaseDifference=0

262
Appendix D: Analogue compensator design software

```matlab
disp('Proportional controller designed. Transfer function: -');
numc = K;
denc = 1;
disp('Compensator');
printsys(numc, denc, 's');
end
disp(' ');

% Display the open loop transfer function of the cascaded system *
% the Nichols-plot of the open-loop cascaded system and the *
% transfer function of the closed-loop system *

disp('Open loop transfer function of plant + compensator:- ');
num_ol_sys = conv(num, numc);
den_ol_sys = conv(den, denc);
printsys(num_ol_sys, den_ol_sys, 's');
disp(' ');

[magr, phaser] = nihols(num_ol_sys, den_ol_sys);
magr = 20*log10(magr);
plot(phaser, magr, 'm');
axis(AX)
title('Nichols Chart of compensated and uncompensated systems');
disp(' ');

disp('Closed-loop transfer function of plant + compensator:- ');
[num_cl_sys, den_cl_sys] = cloop(num_ol_sys, den_ol_sys, -1);
printsys(num_cl_sys, den_cl_sys, 's');
disp(' ');

hold off

% Plotting the closed-loop unit-step responses for the *
% compensated and uncompensated systems *

figure(2)
[NUM, DEN] = cloop(num, den, -1);
[c1, x, t1] = step(NUM, DEN);
[c2, x, t2] = step(num_cl_sys, den_cl_sys);
plot(t1, c1, t2, c2);
t1max = t1(size(t1, 2));
t2max = t2(size(t2, 2));
tmax = t1max;
if t2max > tmax
    tmax = t2max;
end;
t = linspace(0, tmax*2.2, 200);
cl = step(NUM, DEN, t);
c2 = step(num_cl_sys, den_cl_sys, t);
plot(t, cl, t, c2);
title('Unit step responses for compensated & uncompensated systems');
xlabel('Time (sec)');
ylabel('Amplitude');
grid;

% Plotting the closed-loop unit-ramp responses for the *
% compensated and uncompensated systems *

figure(3)
[NUM, DEN] = cloop(num, den, -1);
[c1, x, t1] = step([0 NUM], conv(DEN, [1 0]));
[c2, x, t2] = step([0 num_cl_sys], conv(den_cl_sys, [1 0]));
plot(t1, c1, t2, c2);
t1max = t1(size(t1, 2));
```
Appendix D: Analogue compensator design software

\[
t_{2\text{max}} = t_2(\text{size}(t_2, 2));
\]
\[
t_{\text{max}} = t_{\text{max}};
\]
\[
\text{if } t_{2\text{max}} > t_{\text{max}}
\]
\[
t_{\text{max}} = t_{2\text{max}};
\]
\[
\text{end};
\]
\[
t = \text{linspace}(0, t_{\text{max}}, 200);
\]
\[
c_1 = \text{step}([0 \text{ NUM}], \text{conv}(\text{DEN}, [1 0]), t);
\]
\[
c_2 = \text{step}([0 \text{ num}_1\text{ sys}], \text{conv}(\text{den}_1\text{ sys}, [1 0]), t);
\]
\[
\text{plot}(t, c_1, t, c_2, t, '--');
\]
\[
\text{title}('\text{Unit ramp responses for compensated & uncompensated systems'});
\]
\[
\text{xlabel('Time (sec)')}
\]
\[
\text{ylabel('Amplitude')};
\]
\[
\text{grid};
\]

The above program code may be converted into a MATLAB function using the header line:

```
function [num_cl_sys, den_cl_sys, wc, PhaseP, MagP, numc, denc] = cont_des(num, den, wc);
```
APPENDIX E

MATLAB SOURCE CODE FOR THE
DIGITAL CONTROL DESIGN SOFTWARE

E.1 Main Routine - DIGDES.M

This is the main program code for the single-loop digital controller design tool.

```matlab
% ****************************
% * Digital Controller Design *
% * By Mike A. Oliver        *
% * 12th November 1996.      *
% ****************************
% ****************************************
% * Asks user to enter the numerator and denominator *
% * of the plant transfer function                  *
% ****************************************
clc;
num=input('Enter numerator of plant Transfer Function: ');
den=input('Enter denominator of plant Transfer Function: ');

% ****************************************
% * Draw the Bode plot of the system and prompt the user to *
% * enter the bandwidth of the system                *
% ****************************************
figure(1);
bode(num,den);
disp(' ');
wb=input('From Bode Plot, enter bandwidth of plant: ');
disp(' ');
wwmin=1e-5;
ww=logspace(log10(wwmin),log10(wb),50);

% ****************************************
% * Design an analogue compensator until suitable responses *
% * are found. The design will yield the transfer function of *
% * the continuous closed-loop compensator+plant combination, *
```

---

265
Appendix E: Matlab Source Code For The Digital Control Design Software

```matlab
% * the angular frequency chosen and the required magnitude and * 
% * phase the compensated plot should pass through. * 
% ******************************************************
design=menu('Design Continuous Compensator', ...) 
    'Use Nichols Chart', 'Enter Coefficients'); 
if design==2 
    figure(2);figure(3); 
    NUMC=input('Enter numerator of compensator: '); 
    DENC=input('Enter denominator of compensator: '); 
    [numo,deno]=series(NUMC,DENC,num,den); 
    [num_cl_sys,den_cl_sys]=cloop(numo,deno,-1); 
    MagP=0; 
    [ans,PhaseP,ans,wc]=margin(numo,deno); 
    PhaseP=-(180-PhaseP); 
end; % (* if *) 
while design==1 
    [num_cl_sys,den_cl_sys,wc,PhaseP,MagP,NUMC,DENC]=cont_des(num,den,ww); 
    design=menu('Redesign controller' , 'Yes', 'No'); 
end; %(* while *) 
% ****************************************************** 
% * Get minimum and maximum sampling frequencies * 
% ****************************************************** 
fnNyquist=wc/pi; 
[fmin, fmax]=frqrange(fnNyquist); 
% ****************************************************** 
% * Creating 100 logarithmically distributed frequency and * 
% * sampling interval points * 
% ****************************************************** 
Points=100; 
Fs=logspace(log10(fmin),log10(fmax),Points); 
Ts=1./Fs; 
% ****************************************************** 
% * Obtaining data about the coefficient quantisation * 
% ****************************************************** 
[Qs,bits]=coefdata(0); 
disp(' '); 
% ****************************************************** 
% * Convert plant to the w-plane, design the w-plane * 
% * compensator and convert the continuous controller to * 
% * a z-plane compensator * 
% ****************************************************** 
j=sqrt(-1); 
for i=1:1:Points 
    T=Ts(i); 
    [numw,denw]=s2w(num,den,T); 
    [num_ml_sys,den_ml_sys]=series(num,den,NUMC,DENC); 
    [nump2,denp2]=c2dm(num,den,T,'zoh'); 
    [MagR,PhaseR]=nichols(numw,denw,wc); 
    MagR=20*log10(MagR); 
    PhaseDifference=PhaseP-PhaseR; 
    MagDifference=MagP-MagR; 
    [A,B,numc,dencl]=contdesw(PhaseDifference,MagDifference,wc,T); 
    % ****************************************************** 
    % * Quantising the coefficients * 
    % ****************************************************** 
    if Qs==1 
        [Aq,Bq]=quantfix(A,B,bits); 
    
end; %(* while *)
```

266
elseif Qs==2
[Aq,Bq]=quantfpt(A,B,bits);
else
Aq=A;
Bq=B;
end;

% Determining the complex frequency responses of the *
% continuous closed-loop system, the ideal discrete *
% closed-loop system and the practical discrete *
% closed-loop system.

[Mag,Phase]=bode(num_cl_sys,den_cl_sys,wc/l0);
Phase=Phase*3.1415926536/180;
F=Mag*(cos(Phase)+j*sin(Phase));
[numideal,denideal]=cloop(conv(A,numpz),conv(B,denpz));
G=digfreq(numideal,denideal,wc/l0,T);
[numpract,denpract]=cloop(conv(Aq,numpz),conv(Bq,denpz));
H=digfreq(numpract,denpract,wc/l0,T);

% Determining the error values from the frequency responses *
QCAE(i)=abs(F-H);
AE(i)=abs(F-G);
CRE(i)=abs(G-H);
Freq(i)=1/T;
end

% Plot the error curves *
figure(4);
loglog(Freq,AE,Freq,CRE,Freq,QCAE);
title('Error components at wc');
ylabel('Error Magnitudes');
xlabel('Sampling Frequency (Hz)');

% Prompt the user to select sensible minimum and *
% maximum sampling frequencies based on the error curves *
figure(4);
Ts=clickfrq(0);
Points=size(Ts,2);

% Determining figures of merit for the closed-loop continuous *
% and closed-loop digitally controlled systems.
FOM=zeros(size(Points));
FOMl=zeros(size(Points));
for i=1:1:Points;
T=Ts(i);
wmA=log10(wmin);
wmB=log10(wmax);
numw,denw=s2w(num,den,T);
[MagR,PhaseR]=nichols(numw,denw,wc);
MagR=20*log10(MagR);
PhaseDifference=PhaseP-PhaseR;
MagDifference=MagP-MagR;
Appendix E: Matlab Source Code For The Digital Control Design Software

[A, B, numc, denc] = contdesw(PhaseDifference, MagDifference, wc, T);

% *****************************************************
% * Quantising the coefficients *
% *****************************************************
if Qs == 1
    [Aq, Bq] = quantfix(A, B, bits);
elseif Qs == 2
    [Aq, Bq] = quantfpt(A, B, bits);
else
    Aq = A;
    Bq = B;
end;

% *****************************************************
% * Determining the closed-loop transfer function of the *
% * digital system and calculating the figure of merit. *
% *****************************************************
[numz, denz] = c2dm(num, den, 'zoh');
NUMD = conv(Aq, numz);
DEND = conv(Bq, denz);
[NUMD, DEND] = cloop(NUMD, DEND, -1);
FOM1(i) = figmert2(num_cl_sys, den_cl_sys, NUMD, DEND, T);
FOM(i) = figmert3(num_cl_sys, den_cl_sys, NUMD, DEND, T, ww);
100*i/Points
end

% *****************************************************
% * Finding the digital control system which has the *
% * highest figure of merit. *
% *****************************************************
[ans, best] = max(FOM1);
T = Ts(best);
fltcoef;

% *****************************************************
% * Determining the ADC & DAC wordlength requirements *
% *****************************************************
[ADCword, DACword, quantisation] = adcdac(numc, denc, Aq, Bq, 0.5/T, num, den, T);

% *****************************************************
% * Determining the internal variable requirements *
% *****************************************************
[Overflow, Basic, Underflow] = intvar(Aq, Bq, ADCword);

% *****************************************************
% * Displaying the digital controller requirements *
% *****************************************************
clc;
disp('DIGITAL CONTROLLER REQUIREMENTS');
disp(' ');
disp('Sampling period: T = ' num2str(T) ' seconds');
disp(' ');
disp('Digital transfer function coefficients: ');
disp(' ');
format long;
format compact
Aq
Bq
format;
disp(' ');
disp(' ');

268
Appendix E: Matlab Source Code For The Digital Control Design Software

disp(['Quantisation level: ' num2str(quantisation) ' Volts']);
disp(['ADC wordlength: ' num2str(ADCword) ' Bits']);
disp(['DAC wordlength: ' num2str(DACword) ' Bits']);
disp(' ');
disp(' ');
disp('INTERNAL VARIABLE REQUIREMENTS');
disp(['Basic Wordlength: ' num2str(Basic) ' Bits']);
disp(['Overflow allowance: ' num2str(Overflow) ' Bits']);
disp(['Underflow allowance: ' num2str(Underflow) ' Bits']);

E.2. Function: contdes.m

The source code for contdes.m can be found in Appendix D. However, the line which changes the main routine into a function is required.

E.3. Function: frqrange.m

This function asks the user to enter the minimum and maximum possible values for the sampling frequency. The maximum frequency must be greater than the minimum frequency, which in turn should be greater than the Nyquist frequency.

% FRQRANGE.M
% By Mike A. Oliver 12/11/1996.
% % Asks the user to enter minimum and maximum sampling frequencies
% [MIN, MAX] = FRQRANGE(fNyquist);

function [fmin, fmax] = frqrange(fNyquist);

% ************************************************************************
% * Obtaining the minimum and maximum values for the sampling frequency (Hz) from the user. Checks are used to ensure that
% * the maximum frequency is greater than the minimum frequency
% * and that the Nyquist limit is observed.
% ************************************************************************
disp(' ');
disp(' ');
disp(' ');
disp('Enter the minimum value of sampling frequency (Hz)');
check=0;
while check==0
    disp('Minimum value must be greater than the Nyquist limit');
    fNyquist
    fmin=input('Fmin_');
    if fmin>fNyquist
        check=1;
    end;
end;
check=0;
disp('Enter the maximum value for the sampling frequency (Hz)');
while check==0
    disp('Maximum value must be greater than the minimum value');
    fmax
    fmax=input('Fmax_');
    if fmax>fmin
        check=1;
end;
E.4. Function: coefdata.m

This function uses menu options for the user to select the coefficient quantisation. The options are fixed-point, floating-point or no quantisation. The user is then prompted to enter the coefficient wordlength - the mantissa wordlength for floating-point, and the number of fractional bits for fixed-point.

```matlab
function [Qs,bits]=coefdata(a);

Qs=menu('Coefficient Quantisation Method', 'Fixed-point', 'Floating-point', 'None');
if Qs==1
    ans=menu('Required number of fractional bits', '4', '6', '8', '10', '12', '16', '24', '32');
    if ans==1
        bits=4;
    elseif ans==2
        bits=6;
    elseif ans==3
        bits=8;
    elseif ans==4
        bits=10;
    elseif ans==5
        bits=12;
    elseif ans==6
        bits=16;
    elseif ans==7
        bits=24;
    elseif ans==8
        bits=32;
    end;
elseif Qs==2
    ans=menu('Required number of mantissa bits', '4', '6', '8', '10', '12', '16', '24', '32');
    if ans==1
        bits=4;
    elseif ans==2
        bits=6;
    elseif ans==3
        bits=8;
    elseif ans==4
        bits=10;
    elseif ans==5
        bits=12;
    elseif ans==6
        bits=16;
    elseif ans==7
        bits=24;
    elseif ans==8
        bits=32;
    end;
end;
```
bits=24;
elseif ans==8
  bits=32;
end;
end;

E.5. Function: clickfrq.m

This function requests the user to click on two frequency points on the error versus frequency curves. These two points constitute the minimum and maximum required sampling frequency. The user is then requested to enter the resolution of the timer. A time vector is created starting from the smallest sampling time, increasing by the resolution and ending at the highest sampling time.

```
% CLICKFRQ.M
% By Mike A. Oliver, 12/11/1996.
% Gets the user to click required upper and lower bounds for the sampling frequency and asks for the resolution of changes of T.
% Ts = CLICKFRQ(0);
% '0' is a dummy variable

function Ts = clickfrq(dummy);

% *******************************************************
% * Obtaining the maximum and minimum values for the sampling interval where high performance could be achieved. *
% *******************************************************
clc
hold on
ax=axis;
disp('Use the mouse to click a lower bound on the sampling frequency');
[FreqLow,Mag]=ginput(1);
figure(4);
plot([FreqLow FreqLow],[ax(3) ax(4)],[w--]);
disp(' ');
disp('Use the mouse to click an upper bound on the sampling frequency');
[FreqHigh,Mag]=ginput(1);
figure(4);
plot([FreqHigh FreqHigh],[ax(3) ax(4)],[w--]);
hold off;
Tmin=1/FreqHigh;
Tmax=1/FreqLow;

disp(' ');
deltaT=input('Enter the incremental T step: ');
Tmin=floor(Tmin/deltaT)*deltaT;
Tmax=(floor(Tmax/deltaT)+1)*deltaT;
Ts=Tmin:deltaT:Tmax;
```
E.6. Function: s2w.m

This function simply converts an s-domain transfer function to the w-plane.

```matlab
% S2W.M
% Mike A. Oliver
% 11/05/1996
% Converts an s-domain plant into the w-plane
% [NUMW,DENW]=s2w(num,den,T)
function [numw,denw]=s2w(nums,dens,T);
(numz,denzl)=c2dm(nums,dens,T,'zoh');
[numw,denw]=d2cm(numz,denz,T,'tustin');
```

E.7. Function: contdesw.m

Given the required phase shift and magnitude shift, the controller design frequency and the sampling interval, T, the required compensator is designed and converted to the z-domain.

```matlab
% w-plane Digital Controller Design
% By Mike A. Oliver
% 11th May 1996
% [A,B,numc,denc]=contdesw(PhaseShift,MagShift,wc,T)
function [A,B,numc,denc]=contdesw(PhaseDifference,MagDifference,wc,T);

% ***************************************************
% * Determining the number of cascaded compensators *
% * required *
% ***************************************************
comp=floor(abs(PhaseDifference)/60)+1;
PhaseDifference=PhaseDifference/comp;
MagDifference=MagDifference/comp;
M=10^(MagDifference/20);

% ***************************************************
% * If Phase difference is negative, design a P+I compensator *
% ***************************************************
if PhaseDifference<0
    theta=90+PhaseDifference;
    theta=theta*pi/180;
    Tc=tan(theta)/wc;
    K=M*sin(theta);
    numc=[1];
    denc=[1];
    for i=1:comp
        numc=conv(numc,K*[Tc 1]);
        denc=conv(denc, [Tc 0]);
    end;
```

272
Appendix E: Matlab Source Code For The Digital Control Design Software

end

% */ If phase difference is positive, design a phase advance */
% */ compensator */
% *******************************************************
if PhaseDifference>0
    theta=PhaseDifference*3.1415926536/180;
    k=(l+sin(theta))/(l-sin(theta));
    Tc=1/(wc*sqrt(k));
    G=M/sqrt (k);
    numc=[l];
    denc=[l];
    for i=1:comp
        numc=conv (numc,G*[Tc*k 1]);
        denc=conv (denc,[Tc 1]);
    end;
end

% *******************************************************
% */ If there is no phase difference, design a proportional */
% */ controller */
% *******************************************************
if PhaseDifference==0
    numc=K';
    denc=1;
end

% **************************************
% */ Discretise the w-plane compensator */
% **************************************
[A,B]=c2dm(nurnc,denc,T,'tustin');

E.8. Function: quantfix.m

This is a function which quantises the coefficients of a transfer function to n fractional bits using fixed-point arithmetic.

% QUANTFIX Quantises the coefficients of a digital
% filter whose numerator and denominator
% coefficients are defined by the A and B
% matrices respectively. The coefficients
% are specified to n bits.
% Mike A. Oliver
% [A,B]=QUANTFIX(A,B,n)
function [A,B]=quantfix(A,B,n);

a=size(A);
a=a(2);
b=size(B);
b=b(2);
for c=1:l:a;
    coeff=abs(A(c));

273
quantcoeff=floor((coeff*(2^n))+0.5)/(2^n);
A(c)=sign(A(c))*quantcoeff;
end (%* for *)

for c=1:1:b;
  coeff=abs(B(c));
  quantcoeff=floor((coeff*(2^n))+0.5)/(2^n);
  B(c)=sign(B(c))*quantcoeff;
end (%* for *)

E.9. Function: quantfpt.m

This is a function which quantises the coefficients of a transfer function using floating-point notation. The coefficients are stored using an n-bit mantissa.

% QUANTFPT Quantises the coefficients of a digital
% filter whose numerator and denominator
% coefficients are defined by the A and B
% matrices respectively. The coefficients
% are specified to n bits floating point.
% %
% Mike A. Oliver
% 02/04/1995.
% % [A,B]=QUANTFPT(A,B,n);

function[A,B]=quantfpt(A,B,n);
a=size(A);
a=a(2);
b=size(B);
b=b(2);
for c=1:1:a;
x=(A(c));
  if x==0
    xhat=0;
  else
    if abs(x)<1
      sb=0;
      x1=x;
      while abs(x1)<0.5
        x1=x1*2;
        sb=sb+1;
      end;
      xhat=floor(x1*2^n+0.5)/(2^(n+sb));
    else
      sn=sign(x);
      in=floor(abs(x));
      inb=floor(log(in)/log(2))+1;
      fb=n-inb;
      xhat=sn*floor(abs(x)*2^fb+0.5)/(2^fb);
    end;
  end;
A(c)=xhat;
end (%* for *)

for c=1:1:b;
x=B(c);
end
Appendix E: Matlab Source Code For The Digital Control Design Software

if x==0
   xhat=0;
else
   if abs(x)<1
      sb=0;
      x1=x;
      while abs(x1)<0.5
         x1=x1*2;
         sb=sb+1;
      end;
      xhat=floor(x1*2^n)/(2^(n+sb));
   else
      sn=sign(x);
      in=floor(abs(x));
      inb=floor(log(in)/log(2))+1;
      fb=n-inb;
      xhat=sn*floor(abs(x)*2^fb+0.5)/(2^fb);
   end;
end;
B(c)=xhat;
end %(* for *)

E.10 Function: adcdac.m

This function interacts with the user to determine the ideal parameters for the analogue-to-digital and digital-to-analogue converters. The deterministic error of the digital filter with respect to the w-plane equivalent is plotted with rms estimates of ADC error for various estimates of quantisation level, q. The user is prompted to select the value of q which is most consistent with the deterministic error. The remainder of the routine works out the maximum ADC and DAC voltages and determines suitable wordlengths for the ADC and DAC.

% ADCDAC.M
% By Mike A. Oliver
%
% Interactive function to determine the quantisation level,  
% the ADC and DAC wordlengths for a digital controller that  
% has been designed using w-plane design. 
%
% [ADC, DAC, q]=adc(numc,denc,A,B,fNyquist,num,den,T) 
%
% ADC & DAC are the ADC/DAC wordlengths respectively.  
% q = quantisation level 
% numc,denc = coefficients of the w-plane compensator 
% A, B = coefficients of the digital controller  
% fNyquist = Nyquist frequency of digital controller  
% num,den = plant coefficients  
% T = sampling period.

function [ADCword, DACword,  
   q]=adcdac(numw,denw,Aq,Bq,fNyquist,num,den,T);

% ******************************************************************************
% * Determining a range of signal frequencies *  
% ******************************************************************************

wmax=pi*fNyquist;
wwmin=floor(log10(wmax))end;
ww=logspace(wwmin,log10(wwwmax),100);

275
Appendix E: Matlab Source Code For The Digital Control Design Software

```matlab
figure(5);
F=freqresp(numw,denw,j*ww);
H=freqresp(Aq,Bq,exp(j*ww*T));
Ed=abs(F-H)/sqrt(2); % rms
loglog(ww,Ed,'w');

% Calculating and plotting the deterministic error

% Calculating and plotting the rms estimates of the stochastic error for various values of q
q=0.16;
for i=1:1:5
    q=q/2;
    Ec(i)=quanterr(Aq,Bq,q);
end;
Ecl=[Ec(1) Ec(1)];
Ec2=[Ec(2) Ec(2)];
Ec3=[Ec(3) Ec(3)];
Ec4=[Ec(4) Ec(4)];
Ec5=[Ec(5) Ec(5)];
hold on;
www=[ww(1) ww(100)];
loglog(ww,Ecl,ww,Ec2,ww,Ec3,ww,Ec4,ww,Ec5);

% Determining the ADC and DAC wordlength requirements.
[numz,denz]=c2dm(num,den,T);
[CPnum,CPden]=series(numz,denz,Aq,Bq);
[numADC,denADC]=feedback(1,1,CPnum,CPden);
[numDAC,denDAC]=feedback(Aq,Bq,numz,denz);
t=0:T:T0;
u=ones(size(t));
yadc=filter(numADC,denADC,u);
ydac=filter(numDAC,denDAC,u);
close(5);
maxADC=max(abs(yadc));
maxDAC=max(abs(ydac));
ADCword=floor(log(2*maxADC/q)/log(2))+1;
DACword=floor(log(2*maxDAC/q)/log(2))+1;
```

276
E.11. Function: intvar.m

This function determines the wordlength requirements for the digital controller's internal variables. The basic wordlength is directly determined from the ADC wordlength. The overflow allowance is determined from the coefficients of the digital controller. The underflow requirement is determined by the percentage accuracy specified by the user.

```matlab
% INTVAR.M
% By Mike A.Oliver
% Calculates the Overflow, Underflow and Basic Wordlength
% requirements for a digital filter with known ADC wordlength
% [Overflow, Basic, Underflow] = INTVAR(A,B,ADCword);
function [Overflow, Basic, Underflow] = intvar(A,B,ADCword);

% *******************************************************
% * Check that transfer function is an integrator or not *
% *******************************************************
int=0;
for i=1:size(r,1)
    if r(i)==1
        int=1;
    end; % (* for *)
end; %(* if *)

% ****************************************************
% * Determining the basic wordlength *
% ****************************************************
Basic=ADCword;

% **************************************************
% * Determining the number of overflow bits *
% **************************************************
if int==0
    Overflow=1/sum(B);
elseif int==1
    Overflow=1/sum(A);
end; %(* if *)
Overflow=floor(log(Overflow)/log(2))+1;

% **************************************************
% * Determining the number of underflow bits *
% **************************************************
Underflow=0;
if int==0
    e=input('Enter the required percentage accuracy of the filter: ');
    Underflow=floor(log(100/e)/log(2))+1;
end; %(* if *)
```

Appendix E: Matlab Source Code For The Digital Control Design Software