Computational Algebra and Digital Filter Design

John B. Little

Department of Mathematics and Computer Science

College of the Holy Cross

Worcester, MA 01610

email: little@mathcs.holycross.edu

SIMU '00 Colloquium

July 21, 2000

Outline of Talk

1. DSP fundamentals

2. A digital filter design problem

3. Gröbner bases for solving polynomial systems

4. Results

5. Where next?

Signals

A *signal* is a quantity that depends on time (or space, or any other independent variable(s)).

Examples:

1. a radio transmission

2. an audio waveform

3. an EKG reading

4. the NASDAQ Composite stock index

5. surface motion in an earthquake

6. an image

Signal Processing

Signal processing is the study of operations on signals, for instance:

- breaking signals down into simpler "building blocks" or components to understand how they are put together

- extracting information from signals

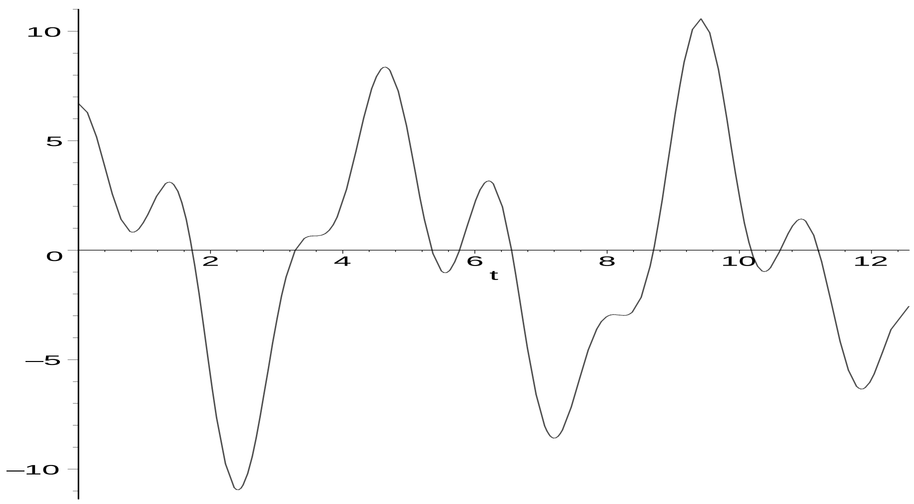
- transforming signals to remove components of high or low frequency

- reducing "noise" components

Analog Signals

In this talk, we consider only signals that can be modeled by scalar functions of one variable (time).

An *analog* signal is a function of a continuous (real) time variable – $f(t)$. For example:

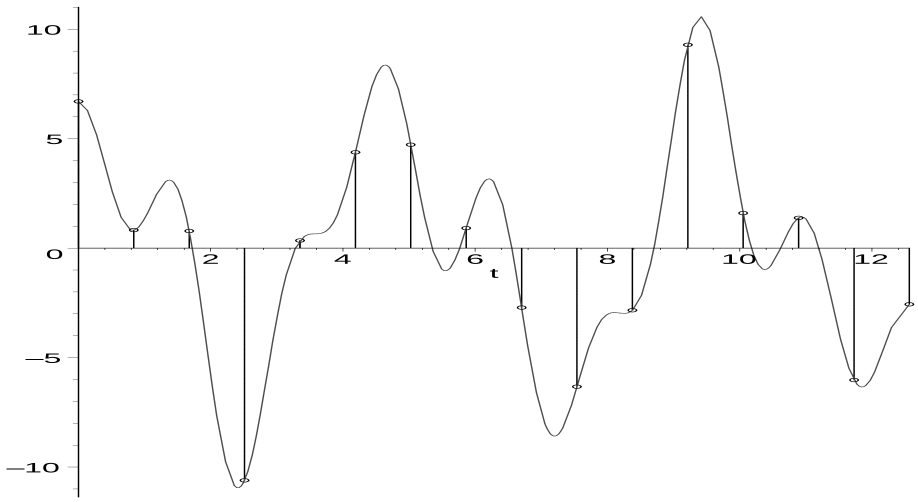


Most "real world" signals are inherently analog in nature.

Digital Signals

A *digital* signal is a function of a *discrete* time variable. That is, we only have values at a discrete set of times $n\Delta t$, n an integer, Δt = "sampling interval". Moreover, the signal values themselves are elements of a discrete set.

Digital signals are often obtained from analog signals by *sampling*



(here $\Delta t = \frac{4\pi}{15}$) and *quantization*.

Why Digital Signals?

Sampling an analog signal always "misses" information about its fine structure.

(However, a theoretical result – the *Sampling Theorem* – says, roughly: *if an analog signal is frequency-limited, then it can be reconstructed exactly from sampled values provided* Δt *is small enough*.)

On the other hand, the availability of fast, inexpensive, programmable digital hardware (computers, special purpose microprocessors, etc.) makes treating signals digitally attractive for many applications where signals are frequency-limited.

Representing Digital Signals

For simplicity, we will mostly ignore quantization effects – *discrete-time, real-valued* signals.

Write a signal as a real-valued function of an integer variable, $x[n]$, or equivalently, a list of real numbers indexed by the integers:

$$\cdots, x[-2], x[-1], x[0], x[1], x[2], \cdots$$

We say a signal is *time limited* if $x[n] = 0$ for all $|n| \geq B$ for some B.

Less restrictive conditions needed for some theoretical results: the ℓ^1 property:

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty$$

or finite energy (ℓ^2):

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$$

Filters

Signal processing operations can be described mathematically by means of functions Γ:

$$\Gamma : \{\text{signals}\} \to \{\text{signals}\}.$$

All of the operations we consider will have the following properties

1. (Linearity) If $z[n] = cx[n] + y[n]$, then

   $$\Gamma(z[n]) = c\Gamma(x[n]) + \Gamma(y[n])$$

2. (Time-invariance) If $y[n] = x[n+k]$ for all n, k fixed, and the outputs are $w[n] = \Gamma(x[n])$, $z[n] = \Gamma(y[n])$, then

   $$z[n] = \Gamma(y[n]) = \Gamma(x[n+k]) = w[n+k]$$

A Γ satisfying both of these properties is said to be an *LTI filter*.

Example – A “Moving Average” Filter

For any input signal $x[n]$, define

$$\begin{aligned} y[n] &= \Gamma(x[n]) \\ &= (x[n] + x[n-1] + x[n-2])/3 \end{aligned}$$

If $x[n] = \cos(n\pi/2)$,

$$\cdots, 1, 0, -1, 0, 1, 0, -1, 0, 1, \cdots$$

for instance, then $\Gamma(x[n]) =$

$$\cdots, 0, 1/3, 0, -1/3, 0, 1/3, 0, -1/3, 0, \cdots$$

It is easy to see that this satisfies the linearity and time-invariance properties, so this defines an LTI filter.

Impulse Response of a Filter

A nice result about LTI filters: They are completely characterized by what they do to *one particular input*.

Let $\delta[n]$ be the signal

$$\cdots, 0, 0, 1, 0, 0, \cdots$$

with the 1 at time $n = 0$ – called the *unit impulse* input.

The output $\Gamma(\delta[n])$ is called the *impulse response* of the filter.

For example, the impulse response of the moving average filter is

$$\cdots, 0, 0, 1/3, 1/3, 1/3, 0, 0, \cdots$$

(the 1/3's occur for $n = 0, 1, 2$; all other values are 0.)

Impulse Response – General Output

Suppose we know the impulse response $h[n] = \Gamma(\delta[n])$ of an LTI filter, and let $x[n]$ be any other signal.

First, note that we can write

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

Hence by linearity

$$\Gamma(x[n]) = \sum_{k=-\infty}^{\infty} x[k]\Gamma(\delta[n-k]).$$

Then by time-invariance

$$\Gamma(x[n]) = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

(Discrete) Convolution

The new signal y obtained by combinining of the signals x, h as above to get

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

is called the *convolution*, written $y = x * h$.

The output of a filter is the convolution of the input and the impulse response.

Filter is *FIR = finite impulse response* if h is time limited – sum is finite then, and we don't need to worry about convergence!

The z-transform

Consider only FIR digital filters from now on.

Package $x[n]$ by its "z-transform", the series

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Do same for the impulse response of a filter $h[n]$ and the output for input $x[n]$:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n}, \ Y(z) = \sum_{n=-\infty}^{\infty} y[n]z^{-n}$$

Coefficient of z^{-n} in the product $H(z)X(z)$ is:

$$\sum_{k=-\infty}^{\infty} h[k]x[n-k],$$

same as $y[n]$ (!)

That is: $Y(z) = H(z)X(z)$–*the z-transform turns convolution into multiplication of series in z.*

Filter Design

A part of what DSP engineers do – design filters to perform various operations on signals, by designing $H(z)$.

Typical problems: design a *low-pass* or *high pass* filters to meet certain specifications.

- Low-pass filters: remove high-frequency components of signals – typically *smooth out* or *blur* signals (can be used to remove high-frequency noise)

- High-pass filters: remove low-frequency components – typically focus on fine details, rapid changes (can be used to detect features)

Fourier Analysis

One standard way to "break down" a signal into simpler components is called Fourier Analysis.

Building blocks: pure frequency sinusoidal signals $x[n] = \cos(an) = \mathrm{Re}(e^{ina})$

General signals are expressed as superpositions of these.

In the digital world, the *discrete Fourier Transform* (DFT) is used. The DFT of $x[n]$:

$$X^F(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-in\omega}$$

The value of $|X^F(\omega)|$ gives the amplitude of the component of x of frequency ω. (The ℓ^1 condition is necessary to make this infinite series converge.)

z- and Fourier Transforms

The z- and Fourier transforms of the impulse response for a filter are almost the same thing!

$$\begin{aligned} H(z) &= \sum_{n=-\infty}^{\infty} h[n] z^{-n} \\ H^F(\omega) &= \sum_{n=-\infty}^{\infty} h[n] e^{-in\omega} \end{aligned}$$

Exact relation: $H^F(\omega) = H(e^{i\omega})$.

Importance of this observation: we can calculate the frequency response characteristics of a filter from its z-transform.

The Magnitude Response

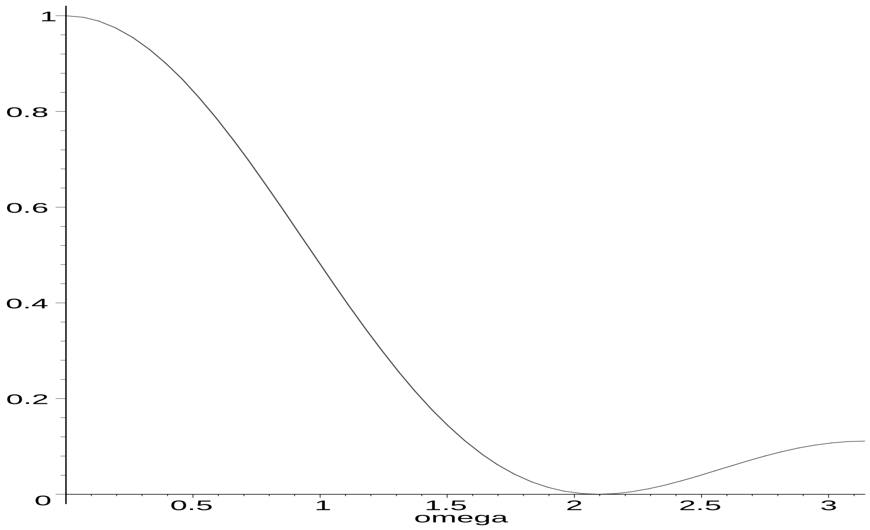
Most important characteristics of a filter are given by its *(squared) magnitude response*

$F(\omega) = |H(e^{i\omega})|^2 = \mathrm{Re}(H(e^{i\omega}))^2 + \mathrm{Im}(H(e^{i\omega}))^2$

and the *group delay*

$$G(\omega) = \frac{d}{d\omega}\Big(\arg(H(e^{i\omega}))\Big)$$

The magnitude response for the moving average filter: $F(\omega) = |(1 + e^{-i\omega} + e^{-2i\omega})/3|^2$



A Low-Pass Filter

For the moving average filter, since $Y(z) = H(z)X(z)$,

1. $\omega > 1.5 \Rightarrow$ the magnitude of the output for an input pure cosine of that frequency will be very small – it gets "filtered out"

2. $\omega < .5 \Rightarrow$ those cosines are being "passed"

In other words – the moving average *is a low-pass filter* (but not a very good one).

Specifications

reference: Selesnick and Burrus, *IEEE Trans. Circuits and Systems II*, v. 45(1), January 1998.

Given the length N of the filter described by

$$H(z) = \sum_{n=0}^{N-1} h[n] z^{-n}$$

provided $N \geq K + L + M + 1$, can find $H(z)$ satisfying:

- $h[0] = 1$ and $F^{(2i)}(0) = 0$, for $i = 1, \ldots M$. (*flatness in passband*).
- $z = -1$ a zero of $H(z)$ of order K (*flatness in stopband* $\omega = \pi$),
- $G^{(2j)}(0) = 0$, for $j = 1, \ldots, L$ (*flat group delay*)

A System of Equations

- Explicit derivatives of F and G give *polynomial equations* in the filter coefficients $h[n]$, or in their *moments*

$$m[k] = \sum_{n=0}^{N-1} n^k h[n], k = 0, 1, \cdots \qquad (1)$$

- $h[n]$ and $m[k]$ are related by the linear equations (1).

- Need to understand the solutions of a fairly large, somewhat messy system of polynomial equations in $m[k]$ – symbolic algebra!

Gröbner Bases

Given $f_1, \ldots, f_s$, polynomials in $\mathbf{R}[x_1, \ldots, x_m]$, if $a \in \mathbf{R}^m$ is any solution of the equations

$$f_1(a) = \cdots = f_s(a) = 0,$$

then any polynomial $f = p_1 f_1 + \cdots + p_s f_s$ also satisfies $f(a) = 0$.

To solve the original system, can try to find "special" combinations of the f_i that give an equivalent, but simpler, system of equations.

Example: Gaussian elimination for systems of linear equations.

A (lexicographic) Gröbner basis is the analog of the triangular form system after Gaussian elimination, but for any system of polynomials (not just linear).

Algorithm for computing them discovered by Buchberger is implemented in computer algebra systems, e.g. Maple.

Shape of a Lex Gröbner Basis

For a system of polynomials $f_1, \ldots, f_s$

1. with only *finitely many* solutions in $\mathbf{C}^m$ (all multiplicity one),

2. whose x_m-components are *distinct*

the "reduced" lex Gröbner basis for $\langle f_1, \ldots, f_s \rangle$ has "triangular" shape:

$$\begin{aligned} g_1 &= x_1 + p_1(x_m) \\ g_2 &= x_2 + p_2(x_m) \\ &\vdots \\ g_{m-1} &= x_{m-1} + p_{m-1}(x_m) \\ g_m &= p_m(x_m) \end{aligned}$$

To solve, find all (real) solutions x_m of $p_m(x_m) = 0$, then substitute to solve for $x_{m-1}, \ldots, x_1$.

Where next?

- This technique gives a good way to control the features of filters – by considering $K + L + M + 1 < N$, we can introduce "tunable" parameters to control the passband cutoff, etc.

- For M large and L small, though, the systems of equations get very complicated! If $L = 0$, the one variable polynomial in the lex Gröbner basis has degree 2^M (!)

- It should be possible to exploit the special structure of these systems though. A close look shows they are nearly triangular already! Work in progress ...

In Conclusion

- Techniques from computational algebra will find more and more use in signal processing for problems like this – they are powerful tools that give different information from other techniques for solving polynomial systems (e.g. iterative techniques like Newton's Method).

- Even the most abstract of mathematics (polynomials, ideals, etc.) will be applied eventually!

- *¡Muchas gracias por su atencíon!*