8

Mathematics for Control

8.1 DIFFERENTIAL EQUATIONS

8.1.1 Breaking Down the State Equations

In Section 6.6, we saw how a system could be described by a matrix state equation of the form

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

in which there are several simultaneous first-order equations.

We have looked at an example where

$$\dot{x} = v$$
$$\dot{v} = bu$$

and we could consider applying feedback

$$u = (-6x - 5v)/b$$

to get

$$\dot{v} = -6x - 5v$$

Essentials of Mechatronics, by John Billingsley
Copyright © 2006 John Wiley & Sons, Inc.

In matrix form these equations are

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \tag{8.1}$$

We can eliminate v from the two equations to get the "conventional" form of a single second-order differential equation

$$\ddot{x} = -6x - 5\dot{x}$$

or

$$\ddot{x} + 5\dot{x} + 6x = 0$$

8.1.2 Solving the Single-Variable Equation

There are a number of ways to solve such an equation. The high-school approach is to say try e^{mt}. If

$$x = e^{mt}$$

then

$$\dot{x} = me^{mt}$$

and

$$\ddot{x} = m^2 e^{mt}$$

so

$$\ddot{x} + 5\dot{x} + 6x = \left(m^2 + 5m + 6\right)e^{mt}$$

The exponential will be nonzero for all finite values of m and t, so we equate the quadratic to zero and solve it, in this case getting roots $m = -2$ and $m = -3$.

The general solution will be

$$x = Ae^{-2t} + Be^{-3t}$$

where A and B are constants determined by the initial conditions

$$x(0) = A + B$$
$$\dot{x}(0) = -2A - 3B$$

8.1.3 Solving the Matrix Equation Directly

Now let us consider the matrix form again. Can we solve Equation (8.1) in a more direct way?

The equation has the form

$$\dot{\mathbf{x}} = A\mathbf{x}$$

Suppose that $\mathbf{x}$ happens to be in the direction of one of the eigenvectors of A, which we will call $\boldsymbol{\xi}$. We could write

$$\mathbf{x} = n\boldsymbol{\xi}$$
$$\dot{\mathbf{x}} = \dot{n}\boldsymbol{\xi}$$

where $\boldsymbol{\xi}$ is a constant vector. Now $A\boldsymbol{\xi} = \lambda\boldsymbol{\xi}$, since that is how eigenvectors and eigenvalues are defined, so $A\mathbf{x}$ will be in the same direction as $\mathbf{x}$;

$$\dot{\mathbf{x}} = A\mathbf{x}$$

tells us that

$$\dot{n}\boldsymbol{\xi} = \lambda n\boldsymbol{\xi}$$

and since $\boldsymbol{\xi}$ is constant, we have

$$\dot{n} = \lambda n$$

which has the solution

$$n = n(0)e^{\lambda t}$$

If λ is positive, this will represent a function that will keep growing to infinity. If λ is negative, it will die away to zero. For stability, this λ must be negative.

But there will be as many eigenvalues and eigenvectors as the order of the system. For second- and higher-order systems, we can express $\mathbf{x}$ as a mixture of the eigenvectors. So now we need *all* the eigenvalues to be negative, since if any one of them should be positive, the corresponding component will grow to infinity.

However, some of the roots could be complex.

Now

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$

So

$$\begin{aligned} e^{(\lambda+j\omega)t} &= e^{\lambda t}e^{j\omega t} \\ &= e^{\lambda t}(\cos(\omega t) + j\sin(\omega t)) \end{aligned}$$

If the real part of the root is positive, the response will be a sine wave that keeps on growing. So, for stability, the real parts of every one of the roots must be negative.

Let us take another look at the response of the position control system, by finding the eigenvalues of the matrix that describes it:

$$\begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix}$$

We take the determinant of $A - \lambda I$

$$\begin{vmatrix} -\lambda & 1 \\ -6 & -5-\lambda \end{vmatrix}$$

and arrive at the quadratic equation

$$\lambda^2 + 5\lambda + 6 = 0$$

The roots are $\lambda = -2$ and $\lambda = -3$. Does that sound familiar?

Replace the m in Section 8.1.2 by s, and you will see something resembling the notation of the Laplace transform. Of course, the roots are the same, yet again.

8.2 THE LAPLACE TRANSFORM

The mathematical justification of the Laplace transform involves integrals over infinite time. The inverse requires an infinite contour integral in the complex frequency domain. But all of this is irrelevant to the way the notation is used by a mechatronic engineer.

8.2.1 The Basis of the Transform

The significant property of its definition is that the transform of the derivative of a function is the variable s times the transform of the function, minus the value of the function at $t = 0$:

$$\mathcal{L}(\dot{x}) = s\mathcal{L}(x) - x(0)$$

This achieves two things. It eliminates derivatives, turning each differentiation into a variable s. It also gives a formal method of dealing with the initial conditions. The result is a function of s for which the corresponding function of time can be looked up in a table. In effect, the table of transforms is a cook book full of "Here's one I prepared earlier."

Now, when we take the transform of our equation for the position system, we get

$$\mathcal{L}(\ddot{x}+5\dot{x}+6x)=(s^2+5s+6)\,\mathcal{L}(x)-sx(0)-\dot{x}(0)-5x(0)$$

so, here is that quadratic again!

With no other input, this expression is equal to zero, so we can rearrange it to get

$$\mathcal{L}(x)=\frac{(s+5)x(0)+\dot{x}(0)}{s^2+5x+6}$$

So now we know the Laplace transform of x, but what is it as a function of time?

The cornerstone of the method is the *uniqueness theorem*, which states that there is one and only one function of time that corresponds to any transform in s. If we have constructed a table of functions and their transforms, then, if we can match the transform, we have found the right function.

In the case above, we can factorize the denominator and split the expression into partial fractions. If, for example, $x(0) = 2$ and $\dot{x}(0) = -5$, we get

$$\mathcal{L}(x)=\frac{1}{s+2}+\frac{1}{s+3}$$

and of course when we look them up in the table, we find the same pair of exponentials as before.

In the mid-1950s, before the Laplace notation became fashionable, the Heaviside D operator was used for the same purpose. Where today we see polynomials in s, then we would have seen polynomials in D, although an extra s appears in the denominators of the functions in the table of transforms.

In the D operator notation, the transform that is just 1 corresponds to the unit step, which is zero for all negative time and has value 1 for all positive time. The Laplace function $1/s$ corresponds to the unit step, but the inverse of the Laplace function 1 is the unit impulse. This has a time integral of 1, but is infinitesimally thin, so that it has to be infinitely tall. It is not a very comfortable function to have to deal with.

8.2.2 Transfer Functions

A useful application of the Laplace transform notation is for the expression of transfer functions. They have an important place in the analysis of control systems, as long as they are not held to be the one and only method.

Consider yet again the motor system described by the equation $\ddot{x} = u$, and suppose yet again that we wish to apply feedback. This time, however, we have no tacho signal and have only x to feed back.

We know that making $u = -ax$ will give us

$$\ddot{x} = -ax$$

which is the equation for simple harmonic motion. Undamped oscillation is not the best kind of control that we might hope for. So, as we did in the experiment of Chapter 3, we try to "guess" the velocity from x.

To estimate the velocity, we first construct x_{slow}, where

$$\frac{d}{dt}x_{\text{slow}} = k(x - x_{\text{slow}})$$

In Laplace terms, ignoring initial conditions, this becomes

$$sX_{\text{slow}} = k(X - X_{\text{slow}})$$

where capitals are used for the transforms, so

$$(s + k)X_{\text{slow}} = kX$$

or

$$X_{\text{slow}} = \frac{k}{s + k}X$$

We estimated the velocity as $k(x - x_{\text{slow}})$ so that

$$\begin{aligned} V_{\text{est}} &= kX - k\frac{k}{s + k}X \\ &= \frac{sk}{s + k}X \end{aligned}$$

So, now that we have an estimated velocity to feed back, let us try

$$\ddot{x} = -ax - bv_{\text{est}}$$

which in Laplace terms is expressed as

$$s^2X = -aX - b\frac{sk}{s+k}X$$

We can multiply through by $(s + k)$ and reorganize to get

$$\left(s^3 + ks^2 + (a+bk)s + ak\right)X = 0$$

To test stability, we look at the roots of the cubic in s. The response will involve terms in e^{st} for each root of the polynomial. As before, if the real part of any root is positive, the exponential will run away and the system will clearly be unstable. So once again we see that all the roots must have negative real parts.

Lemma. A cubic can always has one real root, so it can be factorized into the form

$$\begin{aligned}&(s+p)(s^2+qs+r)\\&\quad = s^3 + (p+q)s^2 + (pq+r)s + pr\end{aligned}$$

Now we know that if and only if p, q, and r are positive, the roots will have negative real parts and the system will be stable. An easy deduction is that the three coefficients of the polynomial must be positive, but there is another condition. Look at the product of the middle two coefficients

$$(p+r)(pq+r)$$

If p, q, and r are positive, this is clearly greater than pr, which is just one of terms when expanded. But this is the product of the first and last coefficients. So, for stability, the product of the middle two coefficients must be greater than the product of the outer two.

In the example above, if a, b, and k are all positive, we can see that the condition for stability is satisfied. So here is a theory confirming that estimating the velocity by this method will always work as far as stability is concerned, but we have to look deeper to select values for the "best performance."

8.2.3 Transfer Functions and Matrices

We can mix the transform method with the matrix state equations, too. When we have

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

we can take the transform to get

$$s\mathbf{X} = A\mathbf{X} + B\mathbf{U}$$

which we can rearrange by introducing a unit matrix I, to get

$$(sI - A)\mathbf{X} = B\mathbf{U}$$

from which we get

$$\mathbf{X} = (sI - A)^{-1} B\mathbf{U}$$

This gives us a *transfer function matrix* that enables us to express each element of **X** in terms of the elements of input function **U**.

8.3 DIFFERENCE EQUATIONS

Until now, when we have used the computer to update the state variables, we have been careful to make the timestep small, so that the approximation to continuous differential equations will be sufficiently accurate. But can we find another way to analyze the system that recognizes the discrete-time nature of computer control?

8.3.1 Sequences of Discrete-Time Samples

As far as the computer is concerned, x is not a continuous function but is defined by a sequence of sampled values, $x_0, x_1, x_2, x_3 \ldots$. The analysis is made much easier if we assume that these are taken at regular equal intervals of time, T, so that our continuous and discrete systems are linked by

$$x_n = x(nT)$$

The computer outputs its control variable u_n very shortly after the measurement of x_n, and u remains constant until the next sample time.

With continuous variables, we defined our equations in terms of rate of change. Now we can instead look at the difference between samples, so that instead of

$$\frac{dx}{dt} = ax + bu$$

we have something like

$$x_{n+1} - x_n = cx_n + du_n$$

but it is all so much simpler if instead of differences we just think of the next value.

$$x_{n+1} = (1+c)x_n + du_n$$

At the end of Section 6.3, we constructed a solution to the differential equation by multiplying both sides by an exponential and integrating. We got Equation (6.4)

$$x(t) = x(0)e^{at} + ub\left(e^{at} - 1\right)/a$$

which calculated the value of x an interval t after applying a constant input u. If the interval is T, we have

$$x(T) = x(0)e^{aT} + u(0)b\left(e^{aT} - 1\right)/a$$

With slight modification this will tell us the value of x at time $(n + 1)T$ in terms x and input u at time nT

$$x((n+1)T) = x(nT)e^{aT} + u(nT)b\left(e^{aT} - 1\right)/a$$

or in terms of our sequence of samples

$$x_{n+1} = x_n e^{aT} + u_n b\left(e^{aT} - 1\right)/a$$

This is in a form similar to that of our original state equation, showing that the next x is a linear combination of the present state and the present input. We could write this as

$$x_{n+1} = ax_n + bu_n$$

but we would risk confusion between the continuous and discrete parameters.

Let us settle for

$$x_{n+1} = px_n + pu_n$$

where

$$p = e^{aT}$$

and

$$q = b\left(e^{aT} - 1\right)/a$$

Now if the input is zero, we obtain

$$x_{n+1} = px_n$$

so

$$x_1 = px_0$$
$$x_2 = p^2 x_0$$
$$x_n = p^n x_0$$

For stability, p^n must not grow indefinitely, so the magnitude of p must not be greater than unity. For a disturbance to decay to zero, we require that

$$|p| < 1$$

8.3.2 Discrete-Time State Equations

We have found a solution for the first-order case, but what if the system is of higher order? Can we use similar methods to solve the matrix differential equation? Can we use

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

to get a discrete-time form?

In Section 6.3, we multiplied both sides by e^{-at} to get an expression that we could integrate. But is there such a thing as e^{-At} when A is a matrix?

We can expand e^{-at} as an infinite series

$$e^{-at} = 1 - at + a^2t^2/2! - a^3t^3/3! \ldots$$

and when we differentiate it term by term, we see a result that is $-a$ times the series with which we started.

In the same way, we can define

$$e^{-At} = I - At + A^2t^2/2! - A^3t^3/3! \ldots$$

and by differentiating term by term, then comparing powers of t against the original series, we see that its derivative is $-e^{-At}A$. So now

$$\frac{d}{dt}\left(e^{-At}\mathbf{x}\right) = e^{-At}\dot{\mathbf{x}} - e^{-At}A\mathbf{x}$$

and by an integral similar to that in Section 6.3, we arrive at

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0) + \left(e^{At} - I\right)A^{-1}B\mathbf{u}$$

when **u** is constant over the interval. Hence

$$\mathbf{x}_{n+1} = e^{At}\mathbf{x}_n + \left(e^{At} - I\right)A^{-1}B\mathbf{u}_n$$

which we can write as

$$\mathbf{x}_{n+1} = P\mathbf{x}_n + Q\mathbf{u}_n$$

The matrix P is the *state transition matrix*, sometimes written as $\Phi(T)$.

With zero input, the state is multiplied by P between samples, so that

$$\mathbf{x}_n = P^n\mathbf{x}_0$$

If λ is an eigenvalue of P, and if $\mathbf{x}_0$ is the corresponding eigenvector, then

$$\mathbf{x}_n = \lambda^n\mathbf{x}_0$$

so if the magnitude of any eigenvalue is greater than unity, the state will run off to infinity. For a disturbance to decay to zero, we require that

$$|\lambda| < 1$$

for every eigenvalue of P.

8.3.3 A Shortcut to Discrete-Time State Equations

For a system like the position controller, there is a more direct way to get the discrete-time state equations. We merely solve the equations in a direct way.

We have

$$\ddot{x} = bu$$

so

$$\dot{x}(t) = \dot{x}(0) + but$$
$$x(t) = x(0) + \dot{x}(0)t + but^2/2$$

We can rewrite these, giving values of x and v at time T, as

$$x(T) = x(0) + v(0)t + ubt^2/2$$
$$v(T) = v(0) + ubT$$

The state equation is therefore

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ v_n \end{bmatrix} + \begin{bmatrix} bT^2/2 \\ bT \end{bmatrix} u_n$$

8.4 THE *Z* TRANSFORM

A mathematician can make the Laplace transform look simple in comparison with the z transform. With contour integrals in the complex frequency plane, summation of infinite series, and an explanation in terms of trains of impulses, the subject can be made somewhat forbidding.

8.4.1 The "Next" Operator

There is, of course, another way to look at the topic. While the Laplace s can be seen as shorthand for d/dt, z can be regarded as meaning "next."

The discrete-time matrix state equation is

$$\mathbf{x}_{n+1} = P\mathbf{x}_n + Q\mathrm{u}_n$$

which we can regard as defining "next" **x**. For the transform, we can write

$$z\mathbf{X} = P\mathbf{X} + Q\mathbf{U}$$

and get a discrete transfer function in the form

$$\mathbf{X} = (zI - P)^{-1} Q\mathbf{U}$$

It is easy to make a connection between the z operator and lines of software. When a variable is changed, we can regard the assignment statement as setting the "next" value.

So, from

```
xslow = xslow + k * (x - xslow) * dt
```

we can replace dt by T and get

$$\mathbf{next}(x_{\text{slow}}) = x_{\text{slow}} + kT(x - x_{\text{slow}})$$

or in transform terms

$$zX_{\text{slow}} = X_{\text{slow}} + kT(X - X_{\text{slow}})$$

so

$$(z-1+kT)X_{\text{slow}} = kTX$$

This gives us the discrete transfer function

$$X_{\text{slow}} = \frac{kT}{z-(1-kT)}X$$

Now V_{est} was given by

$$V_{\text{est}} = k(X - X_{\text{slow}})$$

(there is no extra z because this is "algebra" rather than a state equation)

$$\begin{aligned} V_{\text{est}} &= k\left(1 - \frac{kT}{z-(1-kT)}\right)X \\ &= k\frac{z-1}{z-(1-kT)}X \end{aligned}$$

To work out the transfer function of the double integrator, we look at the final state equation in the previous section:

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_n \\ v_n \end{bmatrix} + \begin{bmatrix} bT^2/2 \\ bT \end{bmatrix}u_n$$

We can write

$$\begin{aligned} (z-1)X &= TV + b(T^2/2)U \\ (z-1)V &= bTU \end{aligned}$$

so, substituting for V and dividing through by $(z - 1)$, we obtain

$$X = \frac{z+1}{(z-1)^2}\frac{bT^2}{2}U$$

Now, if

$$U = -fX - dV_{\text{est}}$$

we can substitute for V_{est} to obtain a polynomial in z multiplying X. The roots of this polynomial will determine whether the system is stable.

As an exercise, try the algebra and see what you can tell about f, d, and kT for stability. There are more conditions to satisfy than in the continuous case.

You can also try *pole assignment*, where you choose three roots that you would like and manipulate the values of f, d, and kT to match the equation coefficients. Try matching three equal roots of 0.5.

The solution is as follows:

$$\begin{aligned} X &= \frac{z+1}{(z-1)^2}\frac{bT^2}{2}U \\ &= \frac{z+1}{(z-1)^2}\frac{bT^2}{2}\left(-fX - dk\frac{z-1}{z-(1-kT)}X\right) \end{aligned}$$

So, multiplying through by the denominators, we have

$$2(z-1)^2(z-(1-kT))X = -f(z-(1-kT))X - dk(z-1)X$$

or bringing everything to the left and taking out the factor X, we obtain

$$\left\{2(z-1)^2(z-(1-kT)) + f(z-(1-kT)) + dk(z-1)\right\}X = 0$$

We end up inspecting the roots of

$$2z^3 - (6-2kT)z^2 + (6-4kT+f+dk)z - (f+dk-fkT) = 0$$

Remember that we are not looking for the simple condition that all the roots have negative real parts, but instead we must show that their magnitudes should all be less than unity.

Instead of struggling, we can "cheat" by saying that we would like three equal roots of 0.5; in other words, the polynomial in z is equivalent to

$$2(z-0.5)^3 = 0$$

(The 2 is there to make the coefficients of z^3 match.)

$$2z^3 - 3z^2 + 1.5z - 0.25 = 0$$

By equating coefficients, we have

$$\begin{aligned} 2kT &= 3, \text{ so } kT = 1.5 \\ f + dk &= 1.5 \\ f + dk - 1.5f &= 0.25 \end{aligned}$$

thus

$$1.5f = 1.25$$

giving

$$f = \frac{5}{6} \quad \text{and} \quad dk = \frac{2}{3}$$

A free decision can still be made concerning the sampling interval.

Do not forget that the values of 0.5 have been pulled out of thin air, without any real justification. The actual behavior of the system might be better assessed by simulation.

8.5 CONVOLUTION AND CORRELATION

Although these seem to be rather abstruse mathematical tricks, heaped with double-summation sigma signs, they are remarkably useful.

8.5.1 Convolution

Having just come to grips with discrete-time control and the z transform, it is appropriate to deal with convolution first.

Let us apply a time function $u(nT)$ to our system. If we wish, we can think of this as a train of outputs to a digital-to-analog converter—there is no need to get tied up with impulses.

Suppose first that we apply just one output, of value 1 at $n = 0$ and zeros from then on. We can express this as a sequence (1,0,0,0, . . .).

How should we describe the output? We are interested only in the sample values at $t = 0$, $t = T$, $t = 2T$, and so on, which we can write as $y(nT)$ or y_n. We might have measured the sequence of values in an experiment or deduced the function from mathematical manipulation of state equations, it does not matter.

So, if we apply the input sequence

$$(1, 0, 0, 0 \ldots.)$$

to our system we have a special *unit response*:

$$(h_1, h_2, h_3, h_4, \ldots, h_n, \ldots)$$

If the first input is of size u_0 instead of 1, we will have an output at each value of nT:

$$y_n = u_0 h_n$$

Now suppose instead that we apply an input at $t = T$, so that

$$u = (0, u_1, 0, 0, 0 \ldots)$$

Everything will happen one sample later, so that the output at $(n + 1)T$ is

$$y_{n+1} = u_1 h_n$$

so

$$y_n = u_1 h_{n-1}$$

In the first case the result of the input had time nT to "mature," but the second input a sample later has only had time $(n - 1)T$ to mature until we sample the output at time nT.

We can go on considering the effect of each individual input u_i at time iT, which will be

$$y_n = u_i h_{n-i}$$

but when we have to consider the effect of the whole input sequence combined, we must add them all up—assuming that the system is linear.

So, now we have an expression with a summation

$$y_n = \sum u_i h_{n-i}$$

Over what range do we have to perform the summation?

Well, it is no use starting before $i = 0$, since we assume that the input sequence started only then. There is no point in continuing beyond $i = n$, unless our system is able to respond to inputs that will happen in the future. (Since we might not always be dealing with time functions, this could sometimes be the case.)

The mathematician would say that y is obtained from the *convolution* of u with h.

In some cases we can regard our system as a filter, which we apply to process the sequence u. It might, for example, be a smoothing filter to present weather data or gasoline prices more neatly. In the cases where we have to perform the summation all the way from the start, it would be called an *infinite impulse response* (IIR) filter, meaning that the effect of a single input will take forever to die away.

But we can use other filters with a limited "window." We might just want to take the average of the latest 10 values, in which case we start summing only from $i = n - 9$. Alternatively, we may consider our lowpass filter to "run out of steam" after the unit response has had 10 intervals to decay, so that we

chop off the sequence at that point to save computing effort. In either case, we will call the filter a *finite impulse response* (FIR) filter.

We will later see this sort of convolution in action in image processing.

8.5.2 Correlation

In convolution, we multiply the terms of one sequence taken left to right by terms from another taken right to left and add up the result.

Correlation is very similar, except the terms are taken in the same direction, but with some displacement between them. So what is it useful for?

Global Positioning System (GPS) satellites transmit a "song" consisting of a repeated *pseudorandom binary sequence* (PRBS). We can think of this as a sequence of +1s and –1s like this:

```
+ + + + – – – + – – + + – + –
```

The 15 symbols repeat to give

```
+ + + + – – – + – – + + – + – + + + + – – – + – – +
```

Now, if we multiply each symbol by itself, we will, of course, get a string of +1s, and if we sum these over a cycle, we will get the answer 15. But what happens if we move the first sequence—let us call it the *template*—relative to the second that we can consider a test sequence. First let us move it by just one symbol:

```
  + + + + – – – + – – + + – + –                 template
+ + + + – – – + – – + + – + – + + + + – –       test
  + + + – + + – – + – + – – – –                 product
```

and now when you sum the terms in the product, you get the answer –1. In fact, you will get this answer when you shift the template relative to the test sequence by any number of symbols except an exact cycle of 15.

Of course, GPS uses much longer sequences, 1023 for *coarse acquisition* and a huge number for the precision signal. However, the principle is the same. By correlating the received signal against a sequence generated in the receiver, it is possible to get a measure of the delay time from satellite to ground—and hence the distance. In fact, each satellite generates a different sequence, and the correlation of one satellite's "song" against another is always near zero.

(If you are interested, the sequence above is $a_3 \oplus a_4$—the next value is 1 if the third to the left is different from the fourth to the left and –1 if they are the same.)

So, we have an expression for the correlation

$$C(n) = \sum a_i b_{i+n}$$

where we sum over the range of the template, a, to give an answer that is a function of the shift, n.

The uses are endless. We can examine an audio signal to try to recognize particular sounds in it. By launching into two dimensions and a double summation, we can examine image data to look for specific objects or characters.

Image correlation is something not to be entered lightly, though. If our template is just 32 pixels square, we have over 1000 multiplications and additions for a single point of the result. But if the test image is 320×320 pixels, we can consider 288 values of shift in each direction. We arrive at some 80 million operations to process a single image, and that is for a specific size and orientation of the template.

We have to take a little care with the template, which will probably contain analog values rather than simple 1s and –1s. We must reduce its mean to zero, so that we will not get a significant response when it is correlated against a constant. We must also smooth its ends, so that the "chopped off" data at the limits of summation do not look like anything of interest.