A discrete-time system is called **shift-invariant** (analogous to time-invariant analog systems ([Time-Invariant Systems (Page 30)](#_bookmark62) ) if delaying the input delays the corresponding output. If *S (x (n)) = y (n)*, then a shift-invariant system has the property



(5.41)

We use the term shift-invariant to emphasize that delays can only have integer values in discrete-time, while in analog signals, delays can be arbitrarily valued.

We want to concentrate on systems that are both linear and shift-invariant. It will be these that allow us the full power of frequency-domain analysis and implementations. Because we have no physical constraints in "constructing" such systems, we need only a mathematical specifcation. In analog systems, the diferential equation specifes the input-output relationship in the time-domain. The corresponding discrete-time specifcation is the **diference equation**.



(5.42)

Here, the output signal *y (n)* is related to its **past** values *y (n - l)*, *l* = {1; … ;*p*}, and to the current and past values of the input signal *x (n)*. The system's characteristics are determined by the choices for the number of coefcients p and q and the coefcients' values {*a*1, ... , *a*p} and {*b*0, *b*1, ... , *b*q}.

ASIDE: There is an asymmetry in the coefcients: where is a0? This coefcient would multiply the *y(n)* term in (5.42). We have essentially divided the equation by it, which does not change the input-output relationship. We have thus created the convention that a0 is

always one.

As opposed to diferential equations, which only provide an **implicit** description of a system (we must somehow solve the diferential equation), diference equations provide an **explicit** way of computing the output for any input. We simply express the diference equation by a program that calculates each output from the previous output values, and the current and previous inputs.

Diference equations are usually expressed in software with for loops. A MATLAB program that would compute the frst 1000 values of the output has the form

for n=1:1000

y(n) = sum(a.\*y(n-1:-1:n-p)) + sum(b.\*x(n:-1:n-q));

end

An important detail emerges when we consider making this program work; in fact, as written it has (at least) two bugs. What input and output values enter into the computation of *y (1)*? We need values for *y (0), y (−1), ...,* values we have not yet computed. To compute them, we would need more previous values of the output, which we have not yet computed. To compute these values, we would need even earlier values, ad infnitum. The way out of this predicament is to specify the system's **initial conditions**: we must provide the p output values that occurred before the