```
Ad =
 0.9048   0         0
    0     0.9489    0.0665
    0     -0.0665   0.9489

Bd =
 0.0952   0
    0     -0.1008
    0     -0.0941

Cd =
    1     0         0
    0     0         1

Dd =
    0     0
    0     -2
```

Let us also find an alternative digital approximation using f.o.h as follows:

```
>>[Ad1,Bd1,Cd1,Dd1] = c2dm(A,B,C,D,0.1,'foh') <enter>

Ad1 =
 0.9048        0         0
    0          0.9489    0.0665
    0          -0.0665   0.9489

Bd1 =
 0.0906        0
    0          -0.1015
    0          -0.0883

Cd1 =
   1    0    0
   0    0    1

Dd1 =
 0.0484        0
    0          -2.0480
```

Note that while the digital state-dynamics matrix, $\mathbf{A_d}$, and the output state coefficient matrix, $\mathbf{C_d}$, are unchanged by changing the hold from z.o.h to f.o.h, the matrices $\mathbf{B_d}$ and $\mathbf{D_d}$ are appreciably modified. This is expected, because the matrices $\mathbf{B_d}$ and $\mathbf{D_d}$ are the coefficient matrices for the input vector, and the hold affects only the sampled, continuous time inputs.

8.9 Solution of Linear Digital State-Equations

A great part of Chapter 4 discussed the solution of the digitized approximation to analog linear, state-equations. Since we have extensively studied the solution of digital state-equations – time varying and time-invariant – in Chapter 4, we shall briefly review the

solution of digital, time-invariant state-equations here. The time propagation of a linear, time-invariant scalar first-order difference equation is directly obtained from the difference equation, namely $x(k+1) = ax(k) + bu(k)$, where $u(k)$ is an arbitrary digital input. Similarly, a linear, time-invariant, digital state-equation is a vector difference equation, expressed as $\mathbf{x}(k+1) = \mathbf{A_d}\mathbf{x}(k) + \mathbf{B_d}\mathbf{u}(k)$, where $\mathbf{u}(k)$ is the digital input vector, and $\mathbf{x}(k)$ is the digital state-vector. Since all the quantities on the right-hand side of a first-order, vector difference equation are known, the value of the state-vector at the next sampling instant, $\mathbf{x}(k+1)$, is directly obtained. MATLAB provides a fast algorithm in the command *ltitr* for the time propagation of a linear, time-invariant digital state-equation, and is used as follows:

```
>>X = ltitr(Ad,Bd,U,X0) <enter>
```

where **Ad** and **Bd** are the state coefficient matrices of the digital system, **U** is a matrix containing the values of the input variables in its columns (each row of **U** corresponds to a different time point), **x0** is the initial condition vector, and **x** is the returned matrix containing values of the state-variables in its columns, with each row corresponding to a different time point. If the initial condition is not specified, *ltitr* assumes zero initial conditions.

Example 8.16

Let us solve the digital state-equation obtained in Example 8.16 with z.o.h, for the first 10 sampling instants, if each of the two input variables is a unit step function, and the initial condition is zero, i.e. $\mathbf{x}(0) = \mathbf{0}$. We begin with specifying the inputs with a sampling interval of $T = 0.2$ second as follows:

```
>>t=0:0.2:2; U=[ones(size(t)); ones(size(t))]'; <enter>
```

Then, the command *ltitr* is used to solve the digital state-equation as follows:

```
>>X = ltitr(Ad,Bd,U) <enter>

X =
   0         0         0
   0.0952   -0.1008   -0.0941
   0.1813   -0.2028   -0.1766
   0.2592   -0.3050   -0.2482
   0.3297   -0.4068   -0.3093
   0.3935   -0.5074   -0.3605
   0.4512   -0.6063   -0.4024
   0.5034   -0.7030   -0.4356
   0.5507   -0.7969   -0.4606
   0.5934   -0.8877   -0.4781
   0.6321   -0.9750   -0.4887
```

Let us also find the response of the digital system obtained in Example 8.15 with f.o.h if the initial condition is $\mathbf{x}(0) = [0.1;\ 1;\ 1]^T$, and the input vector is

$\mathbf{u}(k) = [\sin(kT); -\sin(kT)]^T$. We begin by specifying the inputs as follows:

```
>>t=0:0.2:2; U=[sin(t); -sin(t)]'; <enter>
```

Then, the command *ltitr* is used with the appropriate initial condition as follows:

```
>>X0 = [0.1 1 1]'; X = ltitr(Ad1,Bd1,U,X0) <enter>

X =
  0.1000   1.0000   1.0000
  0.0905   1.0154   0.8824
  0.0999   1.0424   0.7873
  0.1256   1.0810   0.7121
  0.1648   1.1305   0.6536
  0.2141   1.1890   0.6083
  0.2699   1.2541   0.5724
  0.3286   1.3226   0.5420
  0.3866   1.3911   0.5133
  0.4403   1.4556   0.4828
  0.4866   1.5122   0.4472
```

Note that the digital state-space representation allows us to find the response of multivariable digital systems to arbitrary inputs with ease. MATLAB (CST) also provides the commands *dstep*, *dinitial*, and *dimpulse* for the computation of step, initial and impulse response, respectively, of a digital system. The commands *dstep*, *dinitial*, and *dimpulse* are used in a manner similar to the *step*, *initial*, and *impulse* for analog systems, with the difference that the sampling interval is an additional input argument.

For the solution to both Eqs. (8.71) and (8.72) to arbitrary inputs and initial conditions, MATLAB provides the command *dlsim* which is used as follows:

```
>>[y,X] = dlsim(Ad,Bd,Cd,Dd,U,X0) <enter>
```

where **Ad**, **Bd**, **Cd** and **Dd** are the state coefficient matrices of the digital system, **U** is a matrix containing the values of the input variables in its columns (each row of **U** corresponds to a different time point), **x0** is the initial condition vector, **x** is the returned matrix containing values of the state-variables in its columns, with each row corresponding to a different time point, and **y** is the returned matrix containing the output variables at different time points. If the initial condition is not specified, *dlsim* assumes zero initial conditions. Note that while for analog systems, the corresponding MATLAB (CST) command, *lsim*, digitized the continuous time state and output equations before solving them, *dlsim* does not need to do so. For single-input, single-output systems, *dlsim* can be used with pulse transfer function instead of state-space representation (*dlsim* computes a state-space representation internally).

Example 8.17

Let us solve the digital state-equation obtained in Example 8.16 with z.o.h, for the first 10 sampling instants, if each of the two input variables is a *normally distributed random function*, and the initial condition is $\mathbf{x}(0) = [0.1; 1; 1]^T$. We begin with specifying the inputs with a sampling interval of $T = 0.2$ second as follows:

```
>>randn('seed',0); t=0:0.2:2; U=[randn(size(t)); randn(size(t))]' <enter>

U =
   1.1650   -0.7012
   0.6268    1.2460
   0.0751   -0.6390
   0.3516    0.5774
  -0.6965   -0.3600
   1.6961   -0.1356
   0.0591   -1.3493
   1.7971   -1.2704
   0.2641    0.9846
   0.8717   -0.0449
  -1.4462   -0.7989
```

Then, the initial condition is specified and the response is obtained using *dlsim* as follows:

```
>>X0 = [0.1 1 1]'; [y,X] = dlsim(Ad,Bd,Cd,Dd,U,X0) <enter>

y =
   0.1000    2.4023
   0.2013   -1.5436
   0.2418    1.9883
   0.2260   -0.4849
   0.2379    1.2328
   0.1490    0.7274
   0.2962    3.0788
   0.2737    2.9634
   0.4186   -1.5212
   0.4039    0.3430
   0.4484    1.7718

X =
0.1000      1.0000    1.0000
0.2013      1.0861    0.9483
0.2418      0.9681    0.7104
0.2260      1.0303    0.6698
0.2379      0.9640    0.5127
0.1490      0.9852    0.4562
0.2962      0.9788    0.3801
0.2737      1.0902    0.4225
0.4186      1.1907    0.4479
0.4039      1.0604    0.2532
0.4484      1.0276    0.1739
```