PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
IBN-KHALDOUN UNIVERSITY OF TIARET
FACULTY OF APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL ENGINEERING

Laboratory Manual

DIGITAL CONTROL SYSTEMS

Prepared by Dr. OGAB Chahrazad.

Intended for first-year Master students in Embedded Systems Electronics

September 2025

TABLE OF CONTENTS

Foreword

Sampled systems emerged with the digitalization of control systems, particularly when traditional analog control electronics began to be replaced by computers or microprocessors. These digital devices offer numerous advantages, including a wide range of strategies for controller design and computation, the ability to implement more complex and efficient algorithms, and better adaptation to systems that involve delays.

This laboratory manual is intended for first-year Master students specializing in Embedded Systems Electronics. Its goal is to help students understand and study sampled systems using MATLAB. It introduces and guides them in using MATLAB functions and tools related to discrete-time system analysis.

The five lab sessions in this manual progressively cover: the mathematical description of sampled systems, the determination of a discrete linear system's transfer function from recurrence equations, the stability and accuracy of sampled systems, the dynamic behavior of discrete linear systems via state equations, and finally, the analysis and implementation of digital control and observation in the state-space domain.

Through these lab sessions, students will strengthen their theoretical understanding of discrete systems while developing practical MATLAB skills. This manual thus serves as an essential educational resource for training engineers in modern techniques of analysis and control of digital systems.

Introduction to Sampled Systems

1. **Representation of Sampled Systems**

Sampled systems can be represented in various ways. The following summarizes the main modes of representation.

Continuous System		Sampled System
Transfer Function: $G(s)=\dfrac{Y(s)}{U(s)}$		Transfer Function: $\overline{B_0G}(z)=\dfrac{Y(z)}{U(z)}$
State Representation $\dot{\underline{x}}(t) = A\underline{x}(t) + \underline{b}u(t)$ $y(+)=\underline{C}^T\,\underline{x}(+)+du(+)$ (System of n first-order differential equations)	Sampler with Zero-Order Hold	State Representation : $\underline{x}\left[(k+1)^T\right]=F\,\underline{x}(kT)+\underline{g}u(kT)$ $y(kT)=\underline{h}^T\,\underline{x}(kT)+du(kT)$ (System of n first-order recurrence equations)

<u>Transition from Continuous to Discrete State</u>

1.1. Transition from Continuous to Discrete State

The solution to a continuous linear state equation is given by:

$$\underline{x}(+)=e^{At}\underline{x}_0+\int_b^t e^{A(t-z)}bu(z)dZ$$

With $\Phi(+)=e^{At}$ is called the transition matrix

To obtain the discrete state equation, we set $t=kT$ and define:

$$\underline{x}(kT)=e^{Akt}\underline{x}_0+\int_0^{kT} e^{A(kT-z)}\underline{b}u(z)dZ$$

Similarly, we have : $t=(k+1)T$:

$$\underline{x}\left[(k+1)T\right]=e^{A(k+1)T}\underline{x}_0+\int_0^{(k+1)T} e^{A(k+1)T-z)}\underline{b}u(z)dZ$$

this term (1), we obtain

Factoring out e^{AT} in expression of $\underline{x}[(k+1)T]$ we obtain :

$$\underline{x}[(k+1)T] = e^{AT}\left[e^{AkT}\underline{x}_0 + \int_0^{(k+1)T} e^{A(kT-z)}\underline{b}u(z)dz\right]$$

$$= e^{AT}\left[\underbrace{e^{AkT}\underline{x}_0 + \int_0^{kT} e^{A(kT-\tau)}\underline{b}u(\tau)d\tau}_{\underline{x}(kT)} + \int_{kT}^{(k+1)T} e^{A(kT-\tau)}\underline{b}u(\tau)d\tau\right]$$

We perform a change of variable in the second integral to bring it back to the interval [0, T].

$$\alpha = \tau - kT \Rightarrow \tau = kT + \alpha \;\; et \;\; d\tau = d\alpha$$

Hence, we obtain :

$$\underline{x}[(k+1)T] = e^{AT}\underline{x}(kT) + e^{AT}\left[\int_0^T e^{-\alpha A}\underline{b}u(kT+\alpha)d\alpha\right]$$

Assuming the input u(t) remains constant between sampling instants *u(kT+α)=u(kT)* (due to the zero-order hold), we can factor it out of the integral.

$$\boxed{\underline{x}[(k+1)T] = e^{AT}\underline{x}(kT) + e^{AT}\left[\underline{\int_0^T e^{A(T-\alpha)}\underline{b}d\alpha}\right]u(kT)}$$

We set $\begin{cases} F = e^{AT} & \text{which depends only on A} \\ g = \int_0^T e^{A(T-\alpha)}\underline{b}d\alpha & \text{which depends only on A and } \underline{b} \end{cases}$

As well as : $\underline{h}^T = c^T$

This leads to the following discrete-time state-space representation:

$$\begin{cases} \underline{x}[(k+1)T] = F\underline{x}(kT) + \underline{g}u(kT) \\ y(kT) = \underline{h}^T\underline{x}(kT) + du(kT) \end{cases}$$

With :

F : Discrete state transition matrix.

$\underline{g}$: Vector of discrete control input.

$\underline{h}^T$: Observation vector

d : Vector of direct transmission

Example :

Assume zero initial conditions. Consider a second-order continuous-time system whose transfer function is given by:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s(s+a)}$$

From this transfer function, we can determine the following state-space representation:

$$\frac{Y(s)}{U(s)} = \frac{1}{s(s+a)} \Rightarrow sY(s) = \frac{U(s)}{s+a} \quad \Bigg| \quad \begin{array}{l} we\ set: Y(s) = x_1\ ; sY(s) \Rightarrow \dot{y} = \dot{x_1} = x_2 \\ x_2 = \frac{U(s)}{s+a} \Rightarrow \dot{x_2} = -ax_2 + u \end{array}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -a \end{bmatrix} \begin{bmatrix} x_1 \\ x2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(+)$$

$$\text{y(t)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

A discrete-time single-input single-output (SISO) linear system governed by a set of linear recurrence equations with constant coefficients can be represented as:

$$x[(k+1)T] = F\,\underline{x}(kT) + \underline{g}u(kT) \ \ et \ \ y(+) = \underline{h}^T\,\underline{x}(kT)$$

The discrete transition matrix F is obtained by evaluating e^{At} at the sampling period T. The same algebraic methods used in the continuous case (such as Laplace transforms) apply here as well.

Let's go back to the solution of the continuous equation of state:

$$x(t) = e^{At}\underline{x}_0 + \int_0^t e^{A(t-\tau)}\underline{b}u(\tau)d\tau \quad (1)$$

If we use the Laplace transform, we get :

$$\dot{x}(t) = A\underline{x}(+) + \underline{b}u(+) \Rightarrow sx(s) - \underline{x}_0 = Ax(s) + \underline{b}u(s)$$

$$\begin{aligned} sx(s) - Ax(s) &= \underline{x}_0 + \underline{b}u(s) \\ (SI - A)X(s) &= \underline{x}_0 + \underline{b}u(s) \end{aligned} \Rightarrow x(s) = (SI-A)^{-1}\underline{x}_0 + (SI-A)^{-1}\underline{b}u(s) \quad (2)$$

By comparing (1) with (2), we observe that: $e^{At} = \int^{-1}\left((SI-A)^{-1}\right)$

By replacing $t = T \Rightarrow F = e^{AT} = \int^{-1}(SI-A)^{-1}_{t=T}$

$$(SI-A)^{-1} = \frac{1}{\det(SI-A)}.(SI-A)_{adj}$$

$$(SI-A) = S\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & -a \end{bmatrix} = \begin{bmatrix} s & -1 \\ 0 & s+a \end{bmatrix}$$

$$det\ (SI-A) = S(s+a); (SI-A)_{adj} = \begin{bmatrix} s+a & 1 \\ 0 & s \end{bmatrix}$$

$$(SI-A)^{-1} = \frac{1}{s(s+a)}\begin{bmatrix} s+a & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s(s+a)} \\ 0 & \frac{1}{s+a} \end{bmatrix}$$

$$F = e^{AT} = \int (SI-A)^{-1}_{t=T} = \begin{bmatrix} 1 & \frac{1}{a}(1-e^{-at}) \\ 0 & e^{-aT} \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & \frac{1}{a}(1-e^{-at}) \\ 0 & e^{-aT} \end{bmatrix}$$

The vector $\underline{g}$ is obtained from the following relation:

$$\underline{g} = \int_0^T e^{AB}\underline{b}d\beta = \int_0^T \begin{bmatrix} 1 & \frac{1}{a}(1-e^{-a\beta}) \\ 0 & e^{-a\beta} \end{bmatrix} . \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\beta = \int_0^T \begin{bmatrix} \frac{1}{a}(1-e^{-a\beta}) \\ e^{-a\beta} \end{bmatrix} d\beta$$

$$\beta = T - \alpha$$

$$g = \begin{bmatrix} \int_0^T \frac{1}{a}(1-e^{-a\beta}) \\ \int_0^T e^{-a\beta} \end{bmatrix} d\beta = \begin{bmatrix} \frac{1}{a}(T + \frac{e^{-aT}-1}{a}) \\ \frac{1-e^{-aT}}{a} \end{bmatrix}$$

The observation vector is identical to the one corresponding to the continuous system; therefore, we have:

$$\underline{h}^T = \underline{c}^T = [1 \quad 0].$$

1.2. From Discrete Transfer Function to State Equations:

The conversion from a discrete transfer function to state-space representation follows the same procedure as in continuous-time systems.

Let's consider the following example: $\overline{B_0G}(z) = \frac{5z+3}{(z+2)(z+3)} = \frac{Y(z)}{U(z)}$

a. parallel representation:

Decomposing the transfer function into partial fractions gives: $\frac{Y(z)}{U(z)} = \frac{-7}{(z+2)} + \frac{12}{(z+3)}$

By setting: $x_1(z) = \frac{U(z)}{(z+2)}$ *et* $x_2(z) = \frac{U(z)}{(z+3)}$. We then obtain:

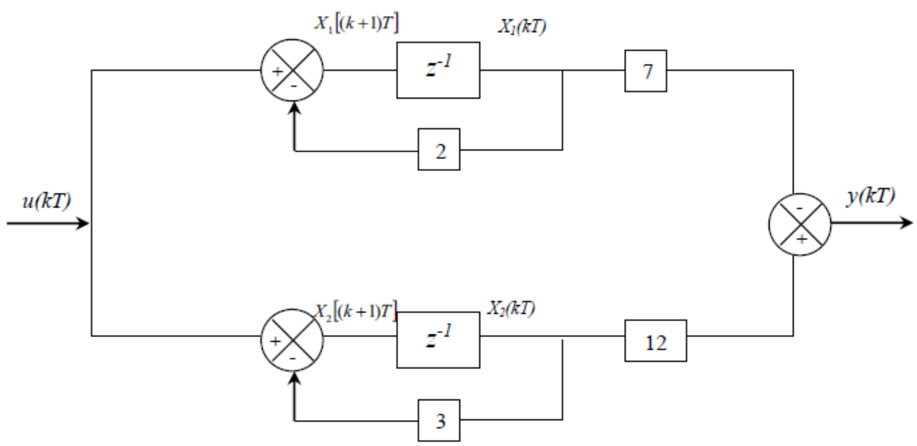
$$\begin{cases} x_1[(k+1)T] = -2x_1(kT) + u(kT) \\ x_2[(k+1)T] = -3x_2(kT) + u(kT) \\ y(kT) = -7x_1(kT) + 12x_2(kT) \end{cases}$$

Hence, the state equations:

$$x[(k+1)T] = \begin{bmatrix} x_1[(k+1)T] \\ x_2[(k+1)T] \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} X_1(kT) \\ X_2(kT) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(kT)$$

$$y(kT) = \begin{bmatrix} -7 & 12 \end{bmatrix} \underline{x}(kT)$$

In simulation graphs, the integration symbols are replaced by the delay operator z^{-1}.



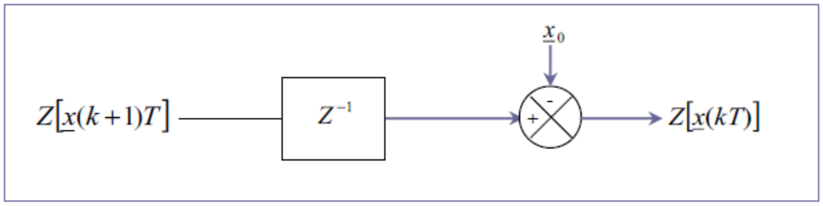
Parallel representation

When initial conditions exist, they must be introduced at the output of each delay operator. This results from the time-shift property in discrete-time systems.

$$Z[(\underline{x}_{k+1})] = Z[z(\underline{x}_k)] = zZ[(\underline{x}_k)] - z\underline{x}_0$$

Then: $Z[(\underline{x}_k)] = z^{-1}Z[x_{k+1}] + \underline{x}_0$

We obtain the result represented by the following diagram:



<u>Initial Conditions in Discrete Systems</u>

1.3. From State Equations to Transfer Function (Direct Computation)

$$X\left[(k+1)T\right] = F\underline{x}\left[(kT) + \underline{g}u(kT)\right]$$

Taking the Z-transform of the discrete state equation gives:

$$z\underline{X}(z) - Z\underline{x}_0 = Fx(z) + \underline{g}u(z)$$

We then have:

$$(ZI - F)\underline{X}(z) = z\underline{x}_0 + \underline{g}u(z)$$

That is : $\underline{X}(z) = (zI - F)^{-1}\underline{x}_0 + (ZI - F)^{-1}\underline{g}u(z)$

We again obtain two terms :

- Transient term : $-\underline{x}(z) = z(zI - F)^{-1}\underline{x}_0$
- Steady-state term : $\underline{x}(z) = z(zI - F)^{-1}\underline{g}u(z)$

For the transfer function, we focus only on the steady-state regime, hence the initial conditions are set to zero. We obtain:

$$\underline{x}(z) = z(zI - F)^{-1}\underline{g}u(z)$$

By taking into account the z-transform of the observation equation, we obtain:

$$\frac{Y(z)}{U(z)} = \overline{B_0 G}(z) = \underline{h}^T(zI - F)^{-1}\underline{g}$$

In the case of a system that is not strictly proper, one simply adds the direct transmission term « d »:

$$\frac{Y(z)}{U(z)} = \overline{B_0 G}(z) = \underline{h}^T (zI - F)^{-1} \underline{g} + d$$

PWNo. 01

Sampling presentation using MATLAB

PWNo. 01 : Sampling Presentation Using MATLAB

1. OBJECTIVE

To discover and learn how to use MATLAB functions and tools related to the study of discrete-time systems.

1. Generation of a discrete-time function
2. Polynomials
3. Computation of the Z-transform and its inverse:
    - Residue method
    - Direct method

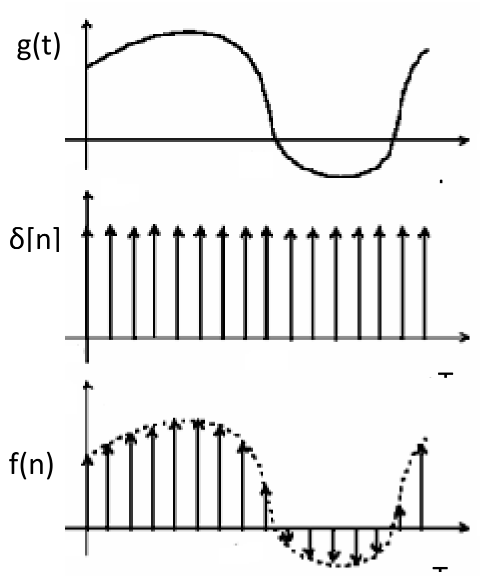
2. Generalities on Discrete Linear Systems:

A discrete signal **f(n)** is a sequence of real numbers called samples. **n** is the index associated with the sampling instant: t=nT. where **T** is the sampling period.

$$f[n] = \sum_{n=0}^{\infty} g[nT]\delta[t-nT]$$

Z-Transform

The Z-transform of the discrete signal f(n) is the function of the variable z, denoted F(z):



$$F(z) = \mathcal{Z}\{f[n]\}$$

$$f[n] = \mathcal{Z}^{-1}\{F(z)\}$$

F(z) is expressed by:

$$F(z) = \sum_{n=0}^{\infty} f[n]z^{-n}$$

Inverse Transform (Time Function)

The inverse transform is given by the following formula:

$$f[n] = \frac{1}{j2\pi}\oint F(z)z^{k-1}dz$$

PART 1

Generation of a Discrete-Time Function

Let x(n)=Acos(wn+thita),with n=0,1,2,.........,N.

Number of simples :N=21; Amplitude A=1; angular frequency w=0.3;

phase :thita=1;

Program 1

```
--- Enter the data----

n=0:N-1;

x=A.*cos(w.*n+thita);

% Plot the function x(n) in the continuous domain

plot(n,x)

% Plot x(n) in the discrete domain

stem(n,x);grid

xlabel('n');ylabel('x(n)')

title('Fonction x(n)')

------------------------------
```

Polynomial (Examples)

- To determine the roots of the polynomial Q=3x^2-3x+2, we use the following program:

Program 2

```
Q=[3 -3 2]

solutions=roots(Q)

------------------------
```

- ✓ To determine the polynomial with roots 1, 2, and 3, where **r** is the vector defining these roots, we use the following program:

Program 3

```
r=[1 2 3]

% The polynomial obtained is:

K=poly(r)

--------------------------
```

PART 2

Z-Transform and Inverse Z-Transform (Examples)

Determine the inverse Z-transform of the function **F(z)=z^3/[(z-0.5)(z-0.75)(z-1)]**

by defining the discrete domain of the system: z and n.

1. Transform the denominator into polynomial form
2. Residue method
3. Direct method

1- Transform the denominator into polynomial form

Program 4 (method 1: we use the **collect** command.)

```
syms n z

% let Dz the denominator of F(z)

Dz=(z-0.5)*(z-0.75)*(z-1);

collect(Dz);
```

Result of the program 4:

collect(Dz)=z^3-9/4*z^2+13/8*z-3/8=1.0000*z^3 -2.2500*z^2+1.6250*z-0.3750 .

- ✓ Verify with your results.

Let **r** be the vector that defines the roots of the denominator Dz (0.5 ; 0.75 ; 1)

Program 5 (method 2: we use the **poly** command)

```
r=[0.5 0.75 1]

% The polynomial obtained is

den=poly(r)
```

Result of the program 5:

den=[1.0000 -2.2500 1.6250 -0.3750];

- ✓ Check against your results

2- Residue Method

Let's now work with F(z)/z

Program 6

```
num=[0 1 0 0];

[num,den]=residue(num,den);

fprintf('\n');

fprintf('r1=%4.2f\t',num(1));fprintf('p1=%4.2f\t',den(1));

fprintf('r2=%4.2f\t',num(2));fprintf('p2=%4.2f\t',den(2));

fprintf('r3=%4.2f\t',num(3));fprintf('p3=%4.2f\t',den(3));
```

Result of the program 6:

r1=8.00; p1=1.00; r2=-9.00; p2=0.75; r3=2.00; p3=0.50;

- ✓ Verify with your results.
- ✓ Deduce the function f(n)(See the theoretical background on the last page of the PW)
- ✓ fn=2*(0.5)^n-9*(0.75)^n+8; Verify with your results.
- ✓ Verify the result by computing the Z-transform of f(n)

Program 7

```
Fz=ztrans(fn,n,z);

simplify(Fz)

%result : ans=8*z^3/(2*z-1)/(4*z-3)/(z-1)

-----------------------------------
```

- ✓ Verify that the inverse of F(z) gives f(n)

Program 8

```
iztrans(Fz)

% Result : ans=2*(1/2)^n-9*(3/4)^n+8;

-------------------------------------------
```

- ✓ Plot the function f(n)for n ranging from 0 to 40

Program 9

```
n=0:40;

fn=2*(1/2).^n-9*(3/4).^n+8;

% plot f(n) in the continuous domain.

plot(n,fn)
```

```
% plot f(n) in the discrete domain.

stem(n,fn);grid

axis([0 60 0 10])

grid
```

- ✓ Use the commands (bar) and (stairs) instead of (plot) and (stem). What do you observe?

PART 3

3- Direct Method

-Direct Calculation of the Z-Transform with Known Function f(n)

Program 10

```
syms n z;fn=3*(-1)^n+6*n-3;Fz=ztrans(fn);simplify(Fz)
```

Result of the program 10 : ans =12*z/(z+1)/(z-1)^2

-Direct Calculation of the Inverse Z-Transform with Known Function F(z)

Program 11

```
syms n z;Fz=12*z/(z+1)/(z-1)^2;fn=iztrans(Fz);simplify(fn)
```

Result of the program 11 : ans =3*(-1)^n-3+6*n

-We calculate the first 20 values of **f(n)** using the (dimpulse) instruction, after expressing the denominator of F(z) in polynomial form.

Program 12(poly)

```
den=poly([-1 1 1]);
```

Result of the program 12 : den=[1 -1 -1 1];

Program 13(collect)

```
collect((z+1)*(z-1)^2)
```

Result of the program 13 :ans=z^3-z^2-z+1

Program 14

```
num=[12 0];den=[1 -1 -1 1];
% Thus, the first 20 values can be computed as follows:
fn=dimpulse(num,den,20)
```

Home Work

1/ Plot the function: y(n)=A.sin(2w.n+thita)+cos(w.n) knowing that: n=0,1,2,....40, A=10, w=314, thita=1

2/ Let a discrete-time system be defined by its Z-transform: (z+1)/[(z-1)(z^2+2z+2)]

- ✓ Compute the **time-domain function f(n)** (inverse Z-transform).
- ✓ Compute the **first 20 values** of f(n).
- ✓ **Plot f(n)** over the interval **[0, 40]**.

Theoretical Background

```
H(z)/z = [r1/(z-p1)]+[r2/(z-p2)+---------]
H(z)= [r1.z/(z-p1)]+[r2.z/(z-p2)]+-----------------
```

***The impulse response** is:*

```
h[n]=r1*(p1^n)*U[n]+r2*(p2^n)*U[n]+------=r1*(p1^n)+r2*p2^n)+--------
```

- To deduce the **step response** (unit step U**(z)**), we first find:

```
Y(z)=H(z).U(z) avec U(z)=z/(z-1)
```

Based on the properties of the Z-transform: The time-domain signal (inverse Z-transform) of `(z/z-a)` is given by: `a^n.U[n]=a^n,` where **U[n]** is the **unit step function.**

PWNo. 02

The Transfer Function of Discrete-Time Systems

PWNo. 02 : The Transfer Function of Discrete-Time Systems

1. OBJECTIVE

- ✓ Determine the transfer function of a discrete linear system from a recurrence equation
- ✓ Determine and plot the system response
- ✓ Simulate the system using Sumilink

PART 1

Plot the impulse response of the system defined by the transfer function:

F(z) = 12*z/((z+1)*(z-1)^2)

Method 1: Determine the expression of the impulse response f(n), and plot f(n) for n=0 to 10.

Program 1

```
syms n z;Fz=12*z/[(z+1)*(z-1)^2];fn=iztrans(Fz);simple(fn)
```

Direct methods for n=0 to 10

- For Methods 2 and 3, one must first determine the vectors (num) and (den) of the transfer function F(z)

- Determine *num* and *den*

```
num=[.   .   .   . ] ; den=[ .   .   .   . ]
```

Method 2: Use the 'dimpulse' command. The syntax for this command in MATLAB is as follows:

```
dimpulse(num,den,10);grid
```

Method 3 with 'impulse'

Program 2

```
% Ts: sampling period

Ts=1

sysd=tf(num,den,Ts,'variable','z^-1')

[y,t]=impulse(sysd)
```

```
impulse(sysd)

%Plot the step response for n=0 to 10

[y,t]=step(sysd,10);

************; grid

-----------------------------
```

- ✓ Simulate the previous responses using a Simulink block.

PART 2

The following equation describes the relationship between the input and output signals of a discrete linear system with initial conditions : **`x[0]=0 , y[0]=0`**

`0.5y[n] - 0.25y[n-1] + 0.0625y[n-2]=0.5x[n]+0.5x[n-1]`

1. Find the transfer function H(z) = Y(z)/X(z)

2. Compute and decompose H(z) = Y(z)/X(z) into partial fractions using the residue method, and then deduce H(z).

Use the instruction below to calculate. : `(r1, p1);(r2, p2);......`

```
fprintf('\n');

disp('r1=');disp(num(1));disp('p1=');disp(den(1));

disp('r2=');disp(num(2));disp('p2=');disp(den(2));
```

3- Plot the impulse response h[n] using direct methods for: n=0:10.

We demonstrate that : `h(n)=[sqrt(2)/4]^n*[cos(n*45°)+5*sin(n*45°)]`using the residue method.

5- Plot the step response (response to a unit step U[n]) using direct methods.

6- Simulate the previous responses using Simulink.

<u>Theoretical Background</u>

The following equation describes the relationship between the input and output signals of a discrete linear system with initial conditions :*x[0]=0 , y[0]=0*

y[n] - 0.5y[n-1] + 0.125y[n-2]=x[n]+x[n-1]

a) By applying the Z-transform to both sides, we obtain:

$Y(z) - 0.5z^{-1}Y(z) + 0.125z^{-2}Y(z) = X(z) + z^{-1}X(z)$

The transfer function is:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1+z^{-1}}{1-0.5z^{-1}+O.125z^{-2}} = \frac{z^2+z}{z^2-0.5z+0.125}$$

b) To obtain the discrete impulse response h(n), we need to calculate the inverse transform of H(z). First, we divide both terms by z, obtaining:

$$H(z)/z = \frac{z+1}{z^2 - 0.5z + 0.125}$$

Using MATLAB's **residue** function, we obtain the residues and poles of $\frac{H(z)}{z}$ as follows:

```
num=…. ;den=……. . ;[num,den]=residue(num,den);

fprintf('\n');

disp('r1=') ;disp(num(1));disp('p1=') ;disp(den(1));…..

disp ('r2=') ; disp(num(2)); disp('p2');disp(den(2));…….
```

r1=0.5000-2.5000i p1=0.2500+0.2500i

r2=0.5000+2.5000i p2=0.2500-0.2500i

and thus,

$$\frac{H(z)}{z} = \frac{0.5 - j2.5}{z - 0.25 - j0.25} + \frac{0.5 + j2.5}{z - 0.25 + j0.25}$$

or

$$H(z) = \frac{(0.5 - j2.5)z}{z - (0.25 + j0.25)} + \frac{(0.5 + j2.5)z}{z - (0.25 - j0.25)} = \frac{(0.5 - j2.5)z}{z - 0.25\sqrt{2}e^{j45^\circ}} + \frac{(0.5 + j2.5)z}{z - 0.25\sqrt{2}e^{-j45^\circ}}$$

Recalling that

$$a^n u_0[n] \Leftrightarrow \frac{z}{z - a}$$

for $|z| > a$, the discrete impulse response sequence h[n] is

$$\begin{aligned} h[n] &= (0.5 - j2.5)(0.25\sqrt{2}e^{j45°})^n + (0.5 + j2.5)(0.25\sqrt{2}e^{-j45°})^n \\ &= 0.5[(0.25\sqrt{2})^n e^{jn45°}] + 0.5[(0.25\sqrt{2})^n e^{-jn45°}] \\ &\quad - j2.5[(0.25\sqrt{2})^n e^{jn45°}] + j2.5[(0.25\sqrt{2})^n e^{-jn45°}] \\ &= 0.5[(0.25\sqrt{2})^n (e^{jn45°} + e^{-jn45°})] - j2.5(0.25\sqrt{2})^n (e^{jn45°} - e^{-jn45°}) \end{aligned}$$

or

$$h[n] = \left(\frac{\sqrt{2}}{4}\right)^n (\cos n45° + 5\sin n45°)$$

c. From $Y(z) = H(z)X(z)$, the transform $u_0[n] \Leftrightarrow \frac{z}{z-1}$, and using the result of part (a) we obtain:

$$Y(z) = \frac{z^2 + z}{z^2 - 0.5z + 0.125} \cdot \frac{z}{z-1} = \frac{z(z^2 + z)}{(z^2 - 0.5z + 0.125)(z-1)}$$

or

$$\frac{Y(z)}{z} = \frac{(z^2 + z)}{(z^2 - 0.5z + 0.125)(z-1)}$$

We will use the MATLAB **residue** function to compute the residues and poles of expression above. First, we must express the denominator as a polynomial.

```
syms z; denom=(z^2–0.5*z+0.125)*(z–1); collect(denom)
ans =
 z^3-3/2*z^2+5/8*z-1/8
```

Then,

$$\frac{Y(z)}{z} = \frac{z^2 + z}{z^3 - (3/2)z^2 + (5/8)z - 1/8}$$

Now, we compute the residues and poles.

```
num=[0 1 1 0]; den=[1 –3/2 5/8 –1/8]; [num,den]=residue(num,den); fprintf(' \n');...
disp('r1 = '); disp(num(1)); disp('p1 = '); disp(den(1));...
disp('r2 = '); disp(num(2)); disp('p2 = '); disp(den(2));...
disp('r3 = '); disp(num(3)); disp('p3 = '); disp(den(3))

r1 =
     3.2000
p1 =
     1.0000
r2 =

   -1.1000 + 0.3000i
 p2 =
    0.2500 + 0.2500i
 r3 =
   -1.1000 - 0.3000i
 p3 =
    0.2500 - 0.2500i
```

With these values, we rewrite $\frac{Y(z)}{z}$ as:

$$\frac{Y(z)}{z} = \frac{z^2+z}{z^3-(3/2)z^2+(5/8)z-1/8} = \frac{3.2}{z-1} + \frac{-1.1+j0.3}{z-0.25-j0.25} + \frac{-1.1-j0.3}{z-0.25+j0.25}$$

or

$$Y(z) = \frac{3.2z}{z-1} + \frac{(-1.1+j0.3)z}{z-0.25-j0.25} + \frac{(-1.1-j0.3)z}{z-0.25+j0.25}$$
$$= \frac{3.2z}{z-1} + \frac{(-1.1+j0.3)z}{z-0.25\sqrt{2}e^{j45°}} + \frac{(-1.1-j0.3)z}{z-0.25\sqrt{2}e^{-j45°}}$$

Recalling that

$$a^n u_0[n] \Leftrightarrow \frac{z}{z-a}$$

for $|z| > a|$, we find that the discrete output response sequence is

$$y[n] = 3.2 + (-1.1+j0.3)(0.25\sqrt{2}e^{j45°})^n - (1.1+j0.3)(0.25\sqrt{2}e^{-j45°})^n$$
$$= 3.2 - 1.1[(0.25\sqrt{2})^n(e^{jn45°} + e^{-jn45°})] + j0.3[(0.25\sqrt{2})^n(e^{jn45°} - e^{-jn45°})]$$

or

$$y[n] = 3.2 - 2.2\left(\frac{\sqrt{2}}{4}\right)^n \cos n45° - 0.6\left(\frac{\sqrt{2}}{4}\right)^n \sin n45°$$
$$= 3.2 - \left(\frac{\sqrt{2}}{4}\right)^n (2.2\cos n45° + 0.6\sin n45°)$$

PWNo. 03

Stability and precision of sampled systems

PWNo. 03 : Stability and precision of sampled systems

2. OBJECTIVE

- ✓ Study of the performance of a discrete-time linear system
- ✓ Preparations for the analysis and correction of discrete-time linear systems

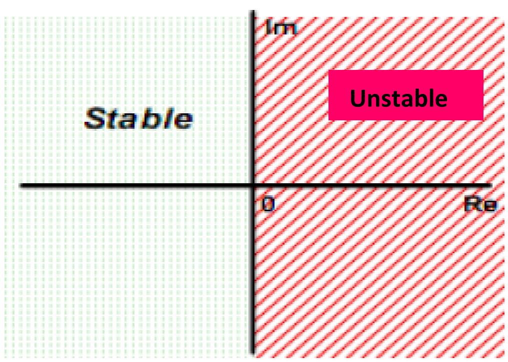
3. Stability of Discrete-Time Systems in the Z-Plane

Similar to continuous-time systems, poles characterize the system dynamics, and the zeros determine its speed of response.

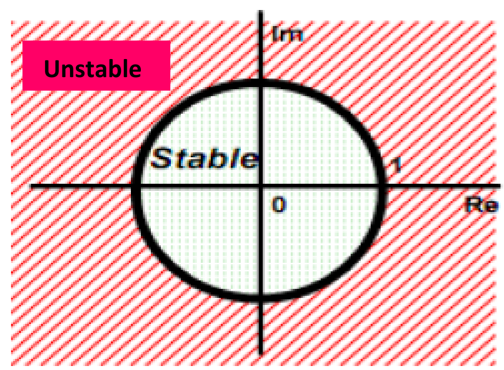
A discrete-time system is said to be stable if its response to a Dirac impulse input converges to zero as time progresses.

The stability condition for continuous-time systems is: **Re (pole) < 0.** Using the continuous-to-discrete mapping relation: ($Z = e^{sT}$). The stability condition for discrete-time systems becomes: **|poles| < 1.** Indeed: $Re(s) < 0 \Leftrightarrow |e^{sT}| + |Z| < 1$

Theorem: A discrete-time system is stable **if and only if** all the poles of its transfer function lie strictly inside the unit circle in the complex Z-plane. The further the poles are from the unit circle (towards the origin), the more damped the system is.



Poles Laplace transform function



Poles Z transform function

PART 1

Example 1

Let us consider the discrete-time system "sysd1", defined by the Open-loop transfer function: G(z)=1/[(z-0.1)(z-0.9)]

- ✓ Plot the root locus and determine the stability limits of the system **sysd1**: (gain, poles, damping, overshoot, frequency (ω))

Program 1

```
Ts=1
```

% System Definition:

sysd1=zpk([],[0.1 0.9],1,Ts)

% root locus:

rlocus(sysd1);zgrid

% Determine the poles and the gain at a selected point on the root locus.

[k,poles]=rlocfind(sysd1)

% Determine the poles, damping ratio, and equivalent frequency (ω) of the open-loop sysd1 system.

damp(sysd1)

Example 2

Determine the stability limits of the system "sysd2" defined by the transfer function F(z)=(z-0.5)/[(z-0.1)(z-0.9)]

Program 2

sysd2=zpk([0.5],[0.1 0.9],1,1)

% Plot the impulse response of both systems, sysd1 and sysd2.

impulse(sysd1,'g');hold on; impulse(sysd2,'r')

- What can be concluded? Explain.

PART 2

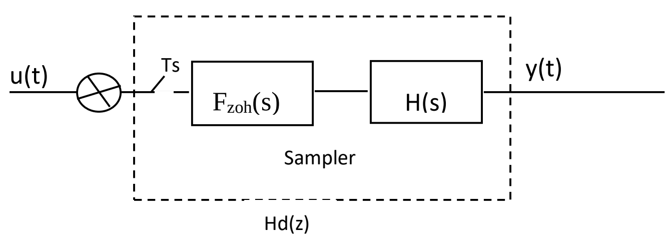
Example

Consider a continuous-time system with the following parameters:

j=0.01;c=0.004;k=10;ki=0.05;

num=ki;

den=[j c k]



- ✓ Determine the continuous-time transfer function H
- ✓ Discretize this continuous-time system using a zero-order hold (F_{zoh})

Recall that:

Hd=Z[F_{zoh}(s)xH(s)]

Hd=[(z-1)/z)].H1

Knowing that: H1=Z(H0)

with: H0=(H/s) and (z-1)/z=(1-z^{-1})

- ✓ Determine H0

Program

```
Ts=0.005;

H0=tf(***********)

H1=c2d(H0,Ts,'zoh')
```

- Determine Hd (Hd=[(z-1)/z)].H1)

```
numd=******

denumd=******

Hd=tf(numd,dnumd,Ts)

% Plot the Bode diagrams of H and Hd

bode(H,'r',Hd,'g')
```

- ❖ Plot the root locus of Hd.
- ❖ Is the system stable? Why?
- ❖ Calculate the poles and the equivalent damping ratio of Hd.

By inserting a compensator D=(z+a)/(z+b), in series with the transfer function Hd; with : a=0.85;b=0

D=zpk(a,b,1,Ts)

- ❖ Determine the equivalent Open-loop transfer function 'Ol'.
- ❖ Compare the frequency responses of 'Hd' and 'Ol' (Bode diagrams).

bode(Hd,'g',Ol,'r')

- ✓ Plot the root locus of the Open-loop system after inserting the compensator and determine the stability limits.
- ✓ What do you observe?
- ✓ Plot the impulse response of Hd and Ol (Open-loop). What conclusions can be drawn?

<u>Analysis of the Closed-loop system.</u>

Cl=feedback(Ol,k); % k: gain placed in the feedback of the Closed-loop system.

- Plot the impulse response of the closed-loop system over the period [0 0.08 0 3e-4].
- Simulate the impulse responses of (Hd and Ol) in Open-loop, and (Cl) in Closed-loop using Simulink.

PWNo. 04

State-Space Representation of Sampled Systems

PWNo. 04 : State-Space Representation of Sampled Systems

1. OBJECTIVE

Describing the dynamic behavior of a discrete-time linear system using state-space equations.

2. Theoretical Background

A linear system can be represented by state-space equations, as illustrated in the table below.

	Continuous system	Discrete-time system
State-space Equation	$\dot{x} = Ax + Bu$	$x[k+1]=Fx[k]+Hu[k]$
Observation Equation	$y=Cx+Du$	$y[k]=Cx[k]+Du[k]$
A ou F	State Transition Matrix or Dynamic Matrix ($n \times n$)	
B ou H	Control Matrix or Input Matrix ($n \times r$)	
C	Observation Matrix or Output Matrix ($p \times n$)	
D	Direct Transmission Matrix ($p \times r$)	

n: Number of state variables = order of the system

r: Number of control inputs

p: Number of outputs

u: Input signal

y: Output signal.

PART 1

- ✓ State-Space Representation of a System:

A=[0 1;0 -1];B=[1;0];C=[1 0];D=0;

- ✓ State-space representation of a continuous-time system (e.g., dc motor)

Motor_c=ss(a,b,c,d)

- Conversion of the system from continuous-time state to discrete-time state (discretization of the continuous system)

Program 1

Ts=1;

motor_d=c2d(motor_c,Ts)

[Ad,Bd,Cd,Dd]=ssdata(motor_d)

PART 2

Consider a discrete system defined by the following recurrence equation:

Y[k+3]+2Y[k+2] +5Y[k+1] +Y[k]=U[k]

U[k] : Input Signal

Y[k] : Output Signal

✓ Write the state equations of the system and determine the matrices: F, H, C, D.

- Use the following as state variables: $X_1[k] = Y[k]$; $X_2[k] = Y[k+1]$; $X_3[k] = Y[k+2]$

Program 2

```
% Sampling period

Ts=0.5;

% Definition of the state-space system - ss( state-space)

sys=ss(F,H,C,D,Ts);

% Plot the responses: step response and impulse response

step(sys,'g');

impulse(sys,'r');grid
```

PART 3

% Determine the transfer function of the previous equation; find the numerator (num) and denominator (den).

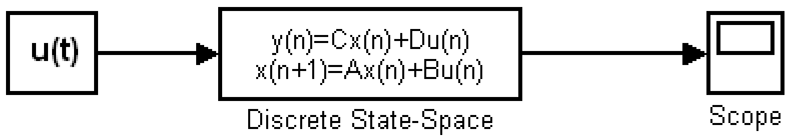
Program 3

```
sys=tf(num,den,Ts);

% Plot the responses: step response and impulse response

step(sys,'g');

impulse(sys,'r');grid
```

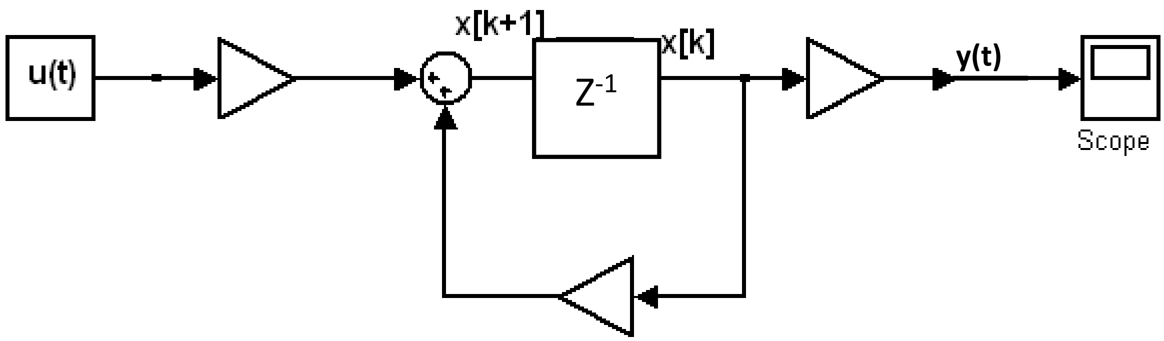
PART 4

- Complete and simulate the following block diagrams
- Plot the step and impulse responses and compare the results

Reduced Model



Detailed Model



PWNo. 05

Control and Observation in Digital State-Space

PWNo. 05 : Control and Observation in Digital State-Space

1. OBJECTIVE

The purpose of this lab work is to analyze and implement a digital control system in the state-space domain. First, a control law will be designed using direct state feedback, and then using estimated state feedback, in order to analyze and compare their performance through simulations in the Matlab/Simulink environment.

2. Practical work:

Consider the system given by the following discrete-time state-space equation (with initial conditions = 0):

$$\begin{cases} x[k+1] = Fx[k] + hu[k] \\ y[k] \quad\;\; = Cx[k] + du[k] \end{cases} \text{With} \quad F = \begin{bmatrix} 0 & 1 \\ -0.368 & 1.368 \end{bmatrix}; \quad h = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; c = [0.264 \quad 0.368]; \quad d=0.$$

Part 1

Create a Simulink file and set the simulation time *Tf* to 60 s:

I Represent the state-space equation using the detailed model (see Lab 4).

II**.** Apply a unit step input **u(k)** and plot the response of the system.

- ✓ Study the controllability and observability of this system using the Matlab commands `ctrb` and `obsv` (see *help ctrb* and *help obsv* for more details).

Part 2

We aim to design a **linear state-feedback control** for this system by assigning the following **closed-loop poles**: **z_1=(0.5-0.5j), z_2=(0.5+0.5j).**

- Use the Matlab function place (see *help* ***place***) to determine the elements of the gain vector **L** for the state-feedback control law **(F−hL).** Then, implement this control in **Simulink** using a detailed model. **e(k)=u(k)-Lx(k).**(see Theoretical Background)
  - At the bottom of **Figure 1**, represent the state-space equation of the system with its control using the detailed model.
  - Keeping the same simulation parameters, plot the system response **before and after applying the control** in the same window. Compare the results.

Part 3

Suppose that x[k] is not accessible, and we want to associate an **observer** with this system, having dynamics characterized by the following **poles**: **z_3=(0.118-0.37j), z_4=(0.118+0.37j);**

- ✓ Determine the elements of the **observer gain vector** v ((F−vC)) using the Matlab function **place**.

- At the bottom of **Figure 2,** represent the state-space equation of the system with its control using the state estimated by the observer (use the detailed model). **e(k)=u(k)-L$\hat{x}$(k).** (see Theoretical Background)
  - ✓ Plot the system response and compare it with the results obtained previously.

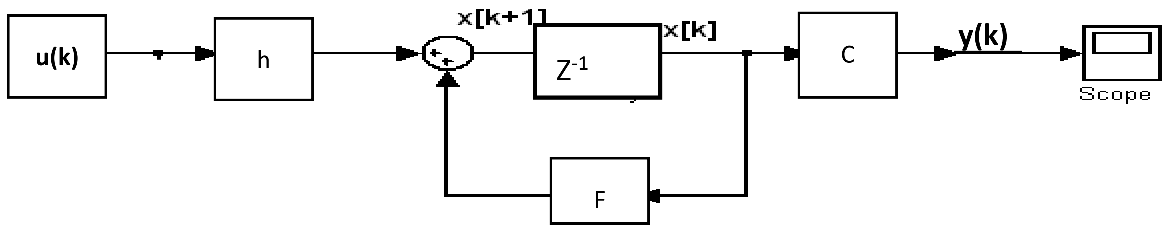


Figure 1

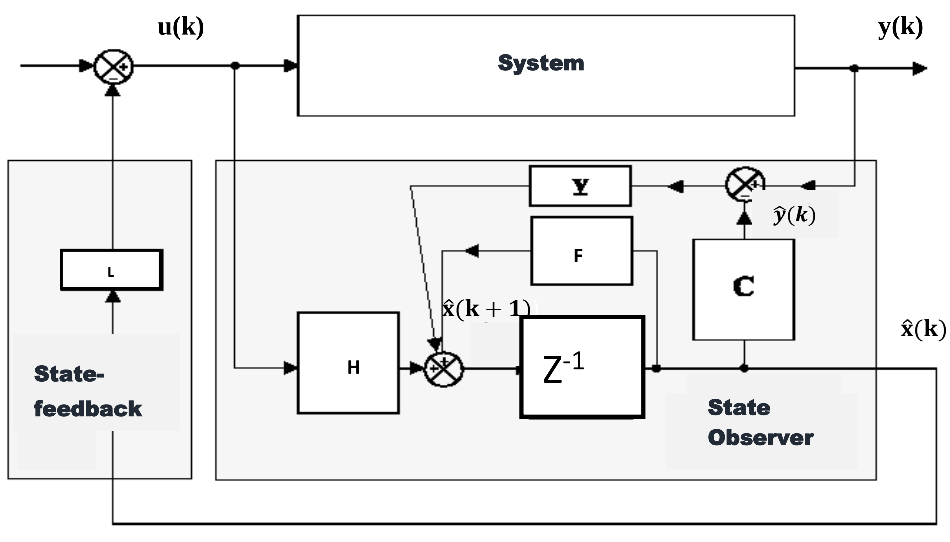


Figure 2

<u>Theoretical Background</u>

- ✓ To compute the elements of the state-feedback gain vector L, the following equation needs to be solved: **det(zI-(F-hL))=(z-z_1)(z-z_2).**
- ✓ To compute the elements of the observer gain vector v, the following equation needs to be solved: **det(zI-(F-vC)=(z-z_3)(z-z_4).**
  - The equation of the observer is given by:

$$\begin{cases} \hat{x}_1(k+1) = -0.396\hat{x}_1(k) + 0.448\hat{x}_2(k) + 1.5y(k), \\ \hat{x}_2(k+1) = -0.896\hat{x}_1(k) + 0.636\hat{x}_2(k) + 2y(k) + u(k). \end{cases}$$

- ✓ Check and validate your results.

References

References:

1. P. Borne. G.D.Tanguv. J. P. Richard. F. Rotella, I. Zambetalcis, "Analyse et régulation de processus industriels-régulation numérique", Tome 2-Editions Technip, 1993.
2. Ogata, K., Solving Control Engineering Problems with MATLAB. Upper Saddle River, NJ: Prentice Hall, 1994.
3. Dorf & Bishop, "Modern Control Systems", Addison-Wesley, 1995.
4. C. Sueur, P. Vanheeeghe, P. Borne, "Automatique des systèmes échantillonnés : éléments de cours et exercices résolus", Technip, 5 décembre 2000.
5. Emmanuel Godoy, Eric Ostertag, "Commande numérique des systèmes: Approches fréquentielle et polynomiale", Ellipses Marketing ,2004.
6. R. Longchamps, "Commande Numérique des systèmes dynamiques", Presse Polytechnique, 2006.
7. Ogata, K., MATLAB for Control Engineers. Upper Saddle River, NJ: Pearson Prentice Hall, 2008.
8. T.J. Katsuhiko, "Modern Control Engineering", fifth Edition, Prentice Hall, 2010.