Demo 3: Simulation of Discrete Systems

In this demo we will examine systems whose dynamics are known only on a discrete set of points. Such systems are usually described by one of more *difference equations*, where the solution takes the form of a *sequence* (as opposed to a continuous function of time). A typical example of a linear first order difference equation would be

$$x(k+1) = 0.7x(k) + p$$

where p represents a parameter. Given an initial condition $x(0)$ and a value for p, the sequence $\{x(0), x(1), \ldots, x(n)\}$ can be computed recursively.

As in the case of continuous systems, here, too, we will be interested in finding *constant solutions* which represent the system equilbria. In our example, such a solution would have to satisfy the equation

$$x^e = 0.7x^e + p$$

which implies that there is a *unique* equilibrium

$$x^e = \frac{p}{0.3}$$

To solve this difference equation in Scilab, we first need to describe the right hand side of the equation in a file which we will call dem3.sci. The contents of this file are shown below.

```
function  y = dem3(t, x, p)
    y=0.7*x+p;
endfunction
```

The Scilab solver requires that we specify an initial condition $x(0)$, an initial time k_0 and a value for parameter p. We also need to indicate the range of points for which we want the solution computed. If we choose $x(0) = 1$, $k_0 = 0$ and $p = 0$, and decide to solve the equation for $k = 0, 1, 2, \ldots, 30$, you should type the following four commands in the workspace:

```
x0 = 1;
k0 = 0;
p = 0;
kvect = 0:1:30;
```

You can then invoke Scilab's difference equation solver with the command

```
y=ode("discrete",x0,k0,kvect,list(dem3,p));
```

Note that this format is very similar to the one we used for solving differential equations. The main difference is the term "discrete", which specifies the nature of the equations.

When plotting the solution, it is convenient to label the computed points as little circles, and to have an additional curve that connects these points (this curve is not necessary, but it often gives us a better sense of how the solution evolves). Such a plot can be obtained using the following command

plot(kvect,y,'.',kvect,y)

where the terms kvect,y,'.' produce the points, and the last two terms create the curve that connects them. The resulting graph is shown in Fig 1.

In analyzing discrete systems, it is often useful to represent $x(k+1)$ as a function of $x(k)$. Observing that

$$y = \begin{bmatrix} x(0) & x(1) & x(2) & \dots & x(29) & x(30) \end{bmatrix}$$

is a vector of dimension 1×31, the commands

w1=y(1:30);

w2=y(2:31);

produce a pair of vectors

$$w_1 = \begin{bmatrix} x(0) & x(1) & x(2) & \dots & x(28) & x(29) \end{bmatrix}$$

and

$$w_2 = \begin{bmatrix} x(1) & x(2) & x(3) & \dots & x(29) & x(30) \end{bmatrix}$$

Typing

plot(w1,w2,'.')

will create pairs $(x(0), x(1))$, $(x(1), x(2))$, ... , $(x(29), x(30))$, which is exactly what we need to display $x(k+1)$ as a function of $x(k)$. The graph obtained in this way is shown in Fig. 2.

To see how the solution changes when we set $p = 2$, type the following sequence of commands:

x0 = 1;

k0 = 0;

p = 2;

kvect = 0:1:30;

and then enter

y=ode("discrete",x0,k0,kvect,list(dem3,p));

as before. The corresponding plot obtained using

plot(kvect,y,'.',kvect,y)

is shown in Fig. 3, which indicates that the solution converges to equilibrium $x^e = 6.667$. Note that this equilibrium is different from the one shown in Fig. 1, due to the fact that p has changed.

Figure 1

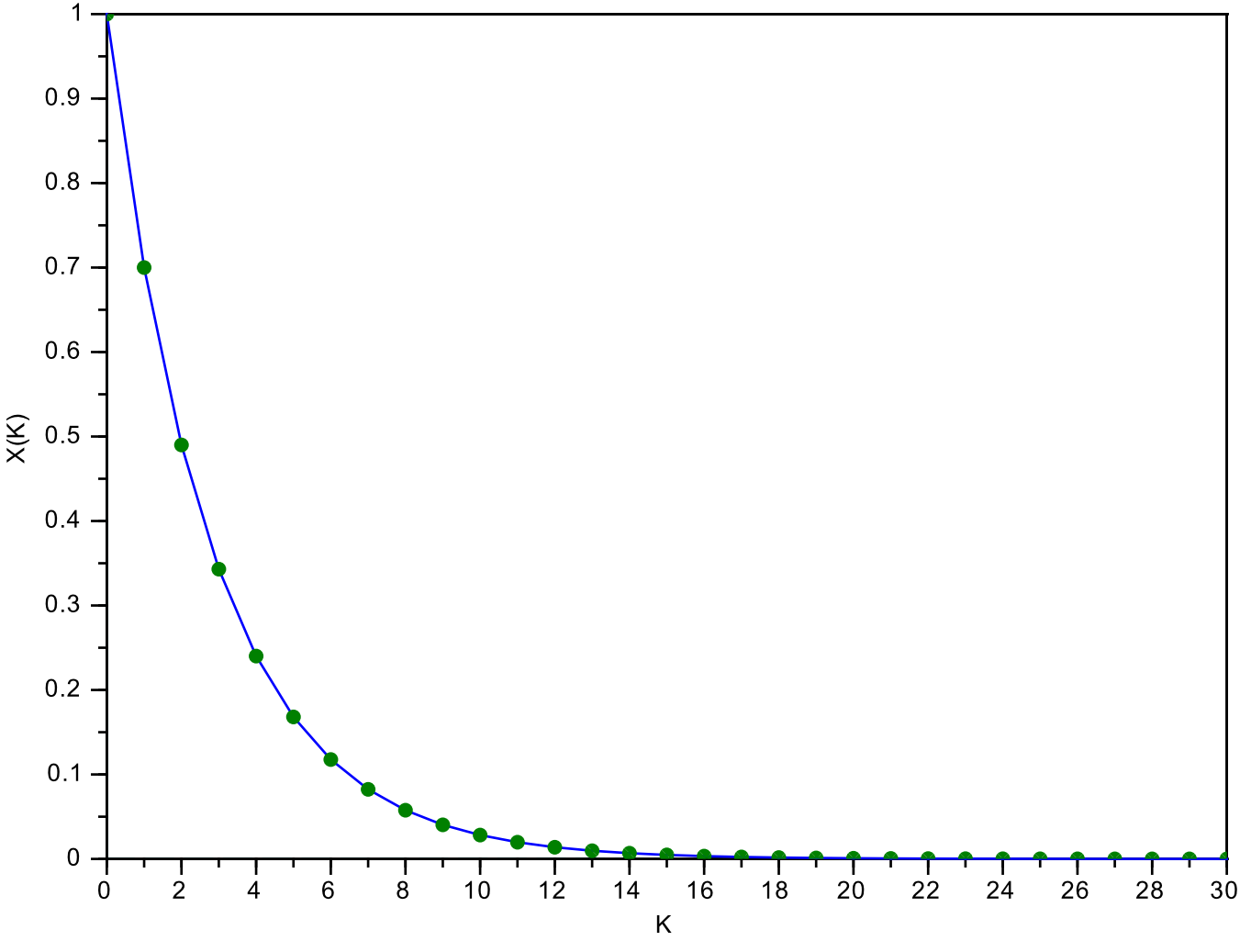

X(K)
K
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

Figure 2

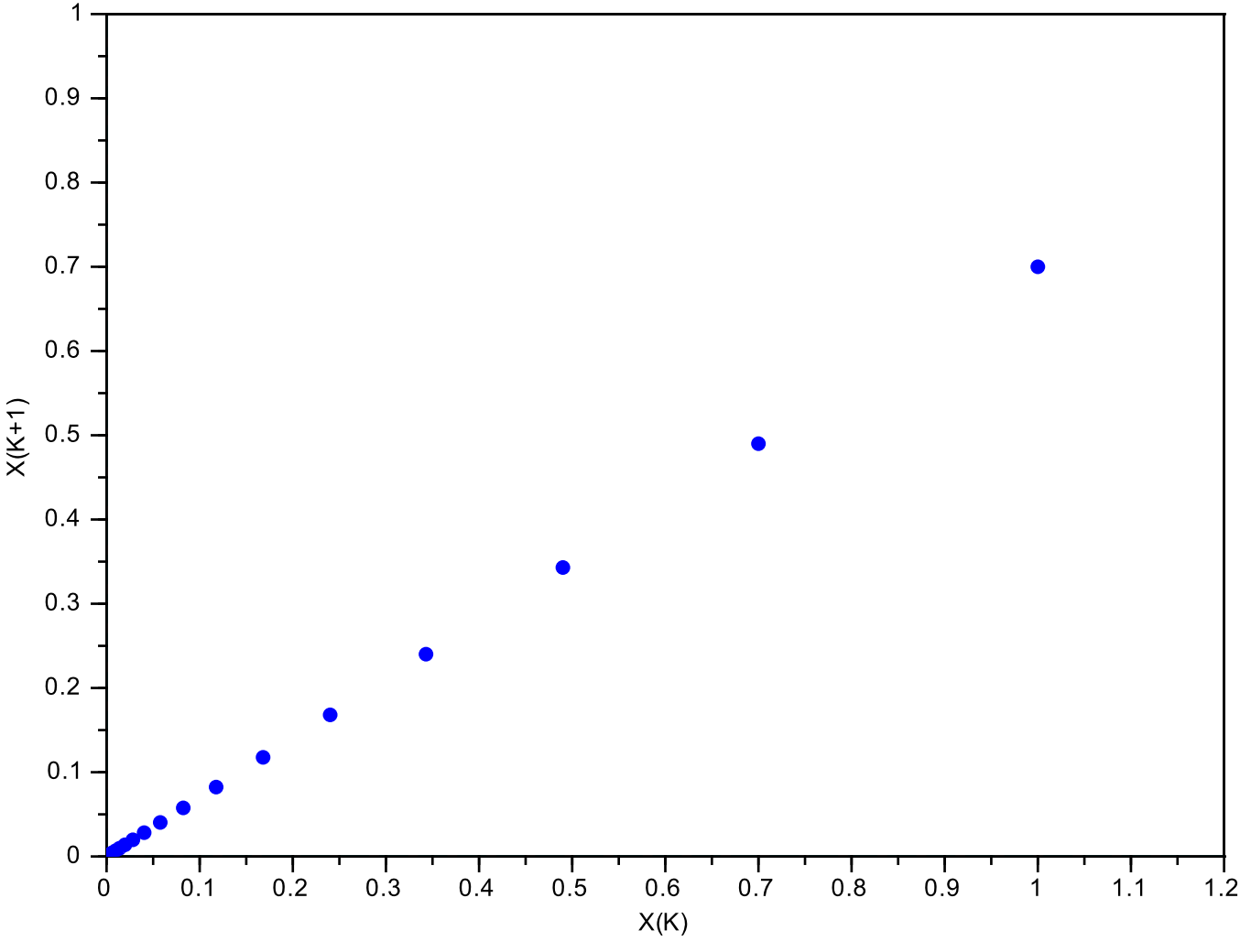
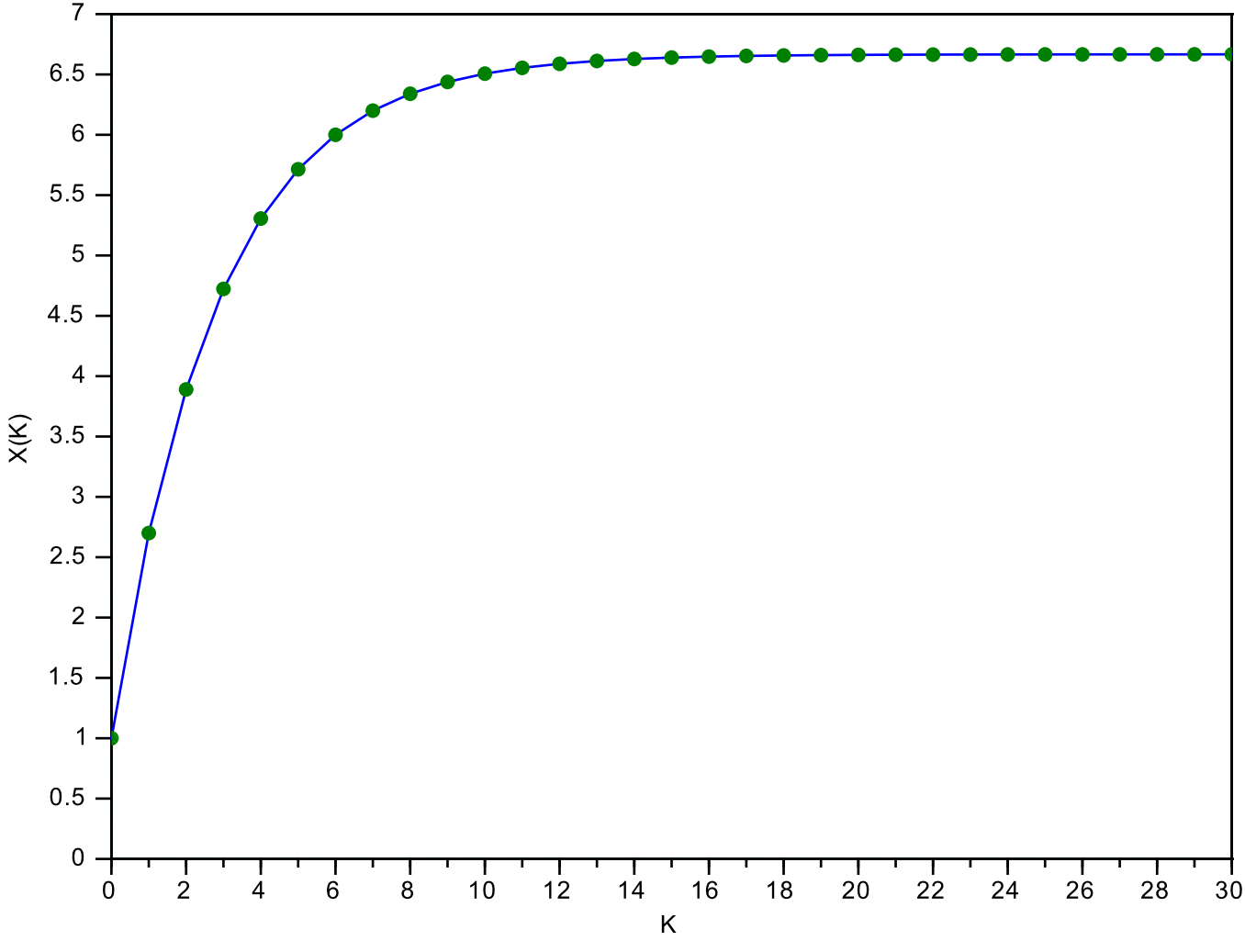


Figure 3


X(K)
K
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30