Mathematical Economics

Alpha Chiang

Chapter 17

Discrete Time: First-Order Difference Equations

17.2 Solving a First-Order Difference Equation

Example 3

Example 4

In [2]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq

from sympy import Function, rsolve
from sympy.abc import t,m,n
y = Function("y");
y0 = Symbol("y_0")
f = m*y(t+1) - n*y(t)  ;
sol = rsolve(f, y(t), {y(0):y0});
sol
```

Out[2]:
$$y_0\left(\frac{n}{m}\right)^t$$

Example 4

In [3]:
```
yt1 = Symbol("y_t+1")
yt = Symbol('y_t')
eq1 = Eq(yt1 - 5*yt,1)
eq1
```

Out[3]:
$$-5y_t + y_{t+1} = 1$$

In [5]:

```python
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y");
f = y(t+1) - 5*y(t) - 1 ;
sol = rsolve(f, y(t), {y(0):7/4});
print("y_t = ${}".format(sol))
display(sol)
```

```
y_t = $2.0*5**t - 1/4
```

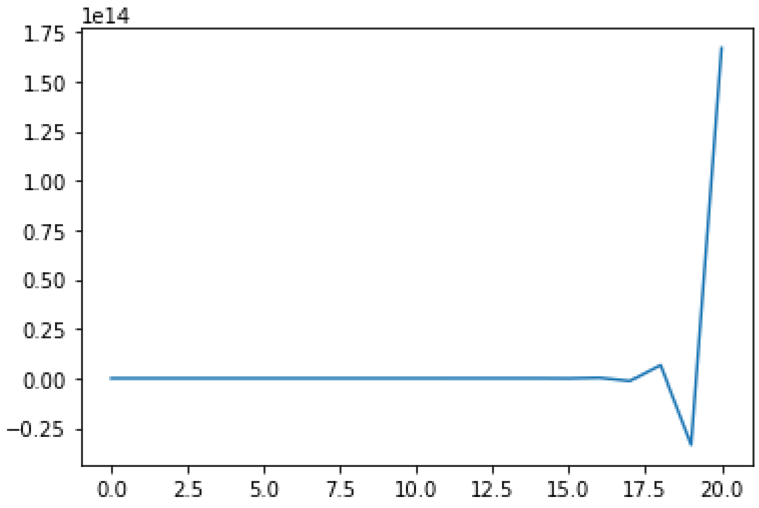
$$2.0 \cdot 5^t - \frac{1}{4}$$

In [6]:

```python
import numpy as np
import matplotlib.pyplot as plt
N = 20
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 7/4
for n in index_set[1:]:
    x[n] = -5*x[n-1]
plt.plot(index_set, x)
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x1fc951660a0>]
```



In [7]:

```python
t = Symbol("t")
yt1 = Symbol("y_t+1")
yt = Symbol('y_t')
```

```python
eq1 = Eq(yt, 2*(-4/5)**t + 9)
eq1
```

Out[7]: $y_t = 2(-0.8)^t + 9$

In [9]:
```python
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y")
f = y(t) - 2*(-4/5)**t - 9
sol = rsolve(f, y(t), {y(0):1})
```

EXERCISE 17.3 --Q3/c--

In [10]:
```python
t = Symbol("t")
yt1 = Symbol("y_t+1")
yt = Symbol('y_t')
eq1 = Eq(yt1 + 1/4*yt, 5)
eq1
```

Out[10]: $0.25y_t + y_{t+1} = 5$

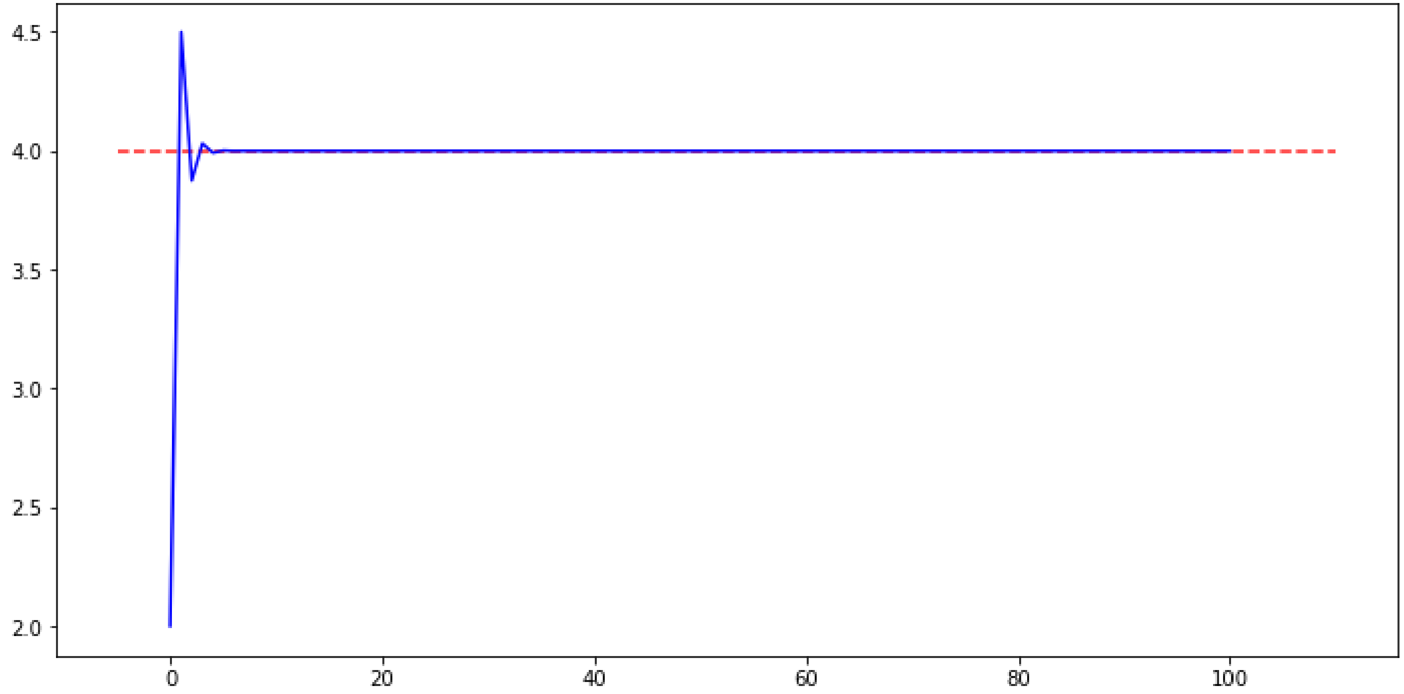
In [11]:
```python
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y")
f = y(t+1) + 1/4*y(t) - 5
sol = rsolve(f, y(t), {y(0):2})
sol
```

Out[11]: $4.0 - 2.0(-0.25)^t$

In [12]:
```python
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 2
for t in index_set[1:]:
    x[t] = -1/4 * x[t-1] + 5

plt.figure(figsize = (12, 6))
plt.hlines(4,-5, 110, linestyle='--', alpha=0.9,color='red')
# Eq value
plt.plot(index_set, x,'b')
```

Out[12]: [<matplotlib.lines.Line2D at 0x1fc954c82e0>]



17.4 The Cobweb Model

In [13]:
```
t = Symbol("t")
Pt1 = Symbol("P_t+1")
Pt = Symbol('P_t')
beta = Symbol('\\beta')
alpha = Symbol('\\alpha')
gamma = Symbol('\\gamma')
delta = Symbol('\\delta')
eq1 = Eq(Pt1 + (delta/beta)*Pt, (alpha+gamma)/beta)
eq1
```

Out[13]:

$$\frac{P_t \delta}{\beta} + P_{t+1} = \frac{\alpha + \gamma}{\beta}$$

In [14]:
```
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y")
P0 = Symbol("P_0")
```

```python
f = y(t+1) + (delta/beta)*y(t) - (alpha+gamma)/beta
sol = rsolve(f, y(t), {y(0):P0})
sol
# For the visulation of Cobweb model
# https://dongminkim0220.github.io/posts/cobweb/
```

Out[14]:

$$\frac{\left(-\frac{\delta}{\beta}\right)^{t}\left(P_{0}\beta + P_{0}\delta - \alpha - \gamma\right)}{\beta + \delta} + \frac{\alpha + \gamma}{\beta + \delta}$$

In [16]:

```python
from sympy import symbols, Eq, solve
t = Symbol("t")
Pt1 = Symbol("P_t+1")
Pt = Symbol('P_t')
Pt1_ = Symbol("P_t-1")
Qd = Symbol("Q_d")
Qs = Symbol("Q_s")

eq1 = Eq(18 - 3*Pt,Qd)
display(eq1)



eq2 = Eq(-3 + 4*Pt1_,Qs)
display(eq2)
eq1.lhs - eq2.lhs
```

$$18 - 3P_{t} = Q_{d}$$

$$4P_{t-1} - 3 = Q_{s}$$

Out[16]: $-3P_{t} - 4P_{t-1} + 21$

In [17]:

```python
eq3 = Eq(-3*Pt -4*Pt1_, -21)
eq3
```

Out[17]: $-3P_{t} - 4P_{t-1} = -21$

In [18]:

```python
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y")
P0 = Symbol("P_0")
```

```
f = -3*y(t) + -4*y(t-1) + 21
sol = rsolve(f, y(t), {y(0):1})
sol
```

Out[18]:

$$3 - 2\left(-\frac{4}{3}\right)^{t}$$

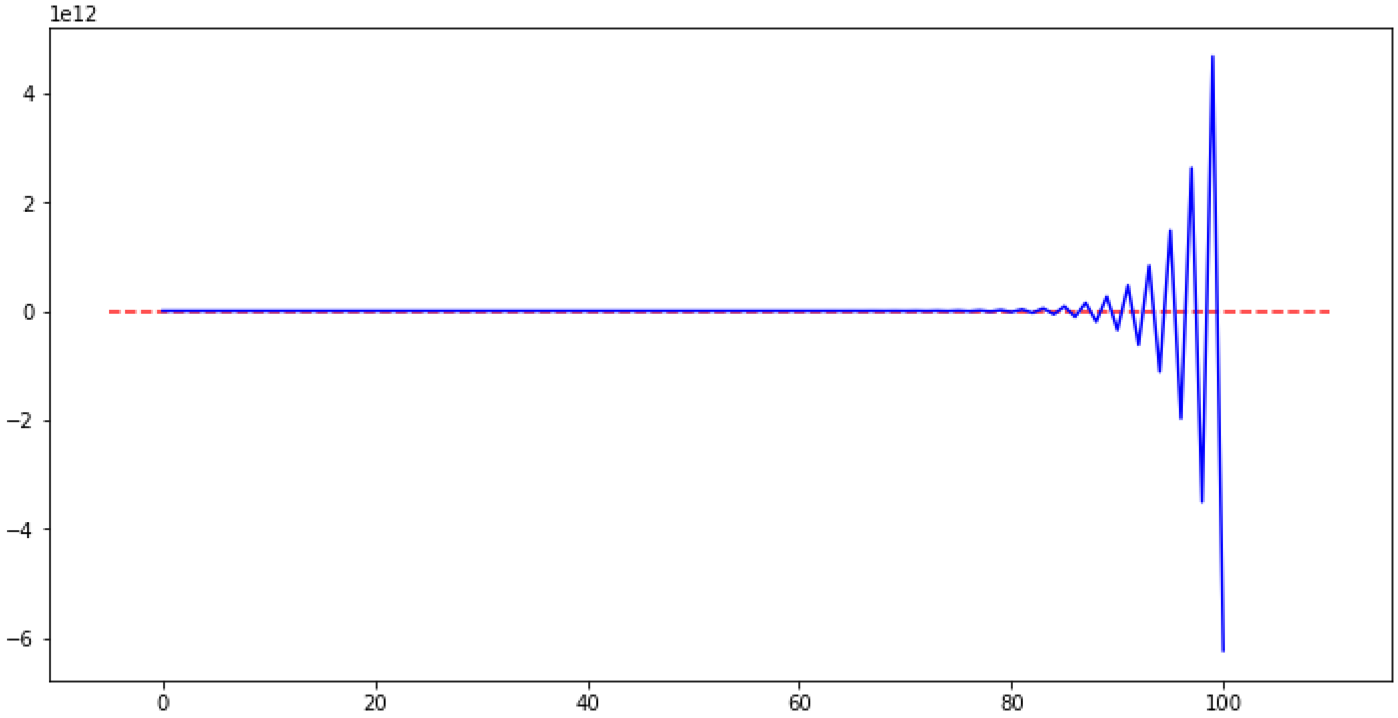
In [19]:

```
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 1
for t in index_set[1:]:
    x[t] = - 4/3 * x[t-1] + 21/3

plt.figure(figsize = (12, 6))
plt.hlines(4,-5, 110, linestyle='--', alpha=0.9,color='red')
# Eq value
plt.plot(index_set, x,'b')
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x1fc9554d8e0>]
```



17.5 A Market Model with Inventory

In [20]:
```
t = Symbol("t")
Pt1 = Symbol("P_t+1")
Pt = Symbol('P_t')
beta = Symbol('\\beta')
alpha = Symbol('\\alpha')
gamma = Symbol('\\gamma')
delta = Symbol('\\delta')
sigma = Symbol('\\sigma')
eq1 = Eq(Pt1 - (1 - sigma*(beta + delta))*Pt,
(alpha+gamma)*sigma)
eq1
```

Out[20]: $-P_t\left(-\sigma\left(\beta+\delta\right)+1\right)+P_{t+1}=\sigma\left(\alpha+\gamma\right)$

In [21]:
```
from sympy import Function, rsolve
from sympy.abc import t
y = Function("y")
```

```
P0 = Symbol("P_0")
f = y(t+1)-(1-sigma*(beta + delta))*y(t)-(alpha+gamma)*sigma
sol = rsolve(f, y(t), {y(0):P0})
sol
```

Out[21]:

$$\frac{(-\beta\sigma - \delta\sigma + 1)^{t}\left(P_{0}\beta + P_{0}\delta - \alpha - \gamma\right)}{\beta + \delta} + \frac{\alpha\sigma + \gamma\sigma}{\sigma\left(\beta + \delta\right)}$$