Mathematical Economics

Alpha Chiang

Chapter 18

Higher-Order Difference Equations

In [1]:
```
from IPython.display import display, Math, Latex
Math(r'\Delta^2 y_{t+1} = \Delta(\Delta y_t) = \Delta(y_{t+1}- y_t)')
```

Out[1]: $\Delta^2 y_{t+1} = \Delta(\Delta y_t) = \Delta(y_{t+1} - y_t)$

18.1 Second-Order Linear Difference Equations with Constant Coefficients and Constant Term

Particular Solution

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

from sympy import Function, rsolve
from sympy.abc import t,c
y = Function("y");
y0 = Symbol("y_0")
a1 = Symbol("a_1")
a2 = Symbol("a_2")

f = y(t+2) + a1*y(t+1) + a2*y(t) - c
sol = rsolve(f, y(t), {y(0):y0});
sol
```

Out[2]:

$$C_0 \left(-\frac{a_1}{2} - \frac{\sqrt{a_1^2 - 4a_2}}{2}\right)^t + \frac{c}{a_1 + a_2 + 1} + \frac{\left(-\frac{a_1}{2} + \frac{\sqrt{a_1^2 - 4a_2}}{2}\right)^t \left(-C_0\left(a_1 + a_2 + 1\right) + a_1 y_0 + a_2 y_0 - c + y_0\right)}{a_1 + a_2 + 1}$$

Example 1

In [3]:
```
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
eq1 = Eq(yt2 - 3*yt1 +4*yt,6)
display(eq1)

from sympy.abc import t,c,k
y = Function("y");


f = y(t+2) - 3*y(t+1) + 4*y(t)-6;
sol = rsolve(f, y(t), {y(0):1, y(1):2});
sol
```

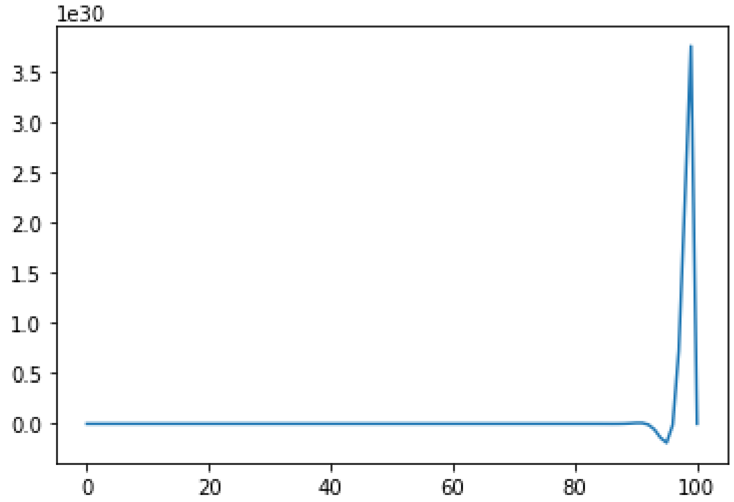
$$4y_t - 3y_{t+1} + y_{t+2} = 6$$

Out[3]:

$$\left(-1 + \frac{2\sqrt{7}i}{7}\right)\left(\frac{3}{2} - \frac{\sqrt{7}i}{2}\right)^t + \left(-1 - \frac{2\sqrt{7}i}{7}\right)\left(\frac{3}{2} + \frac{\sqrt{7}i}{2}\right)^t + 3$$

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 1
x[1] = 1
for t in index_set[1:N]:
    x[t] = 3*x[t-1] - 4*x[t-2] + 6
plt.plot(index_set, x)
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x214dc5ae070>]
```



Example 2

In [5]:
```
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
eq1 = Eq(yt2 + yt1 - 2*yt, 12)
display(eq1)



from sympy.abc import t,c,k
y = Function("y");


f = y(t+2) + y(t+1) - 2*y(t)- 12;
sol = rsolve(f, y(t), {y(0):1, y(1):2});
sol
```

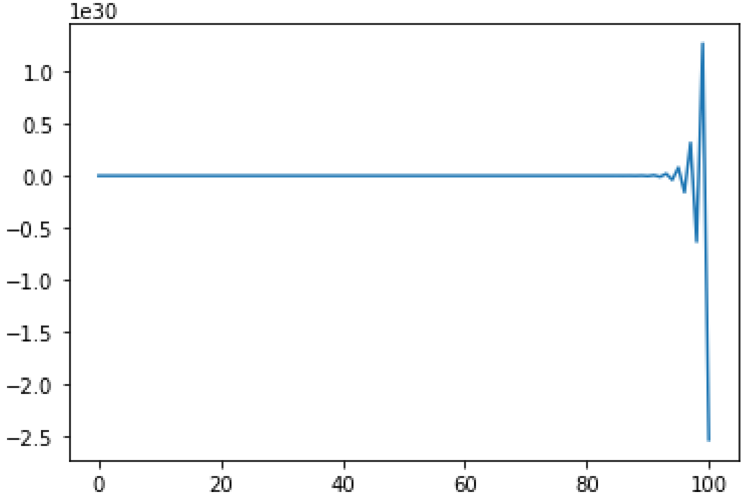
$$-2y_t + y_{t+1} + y_{t+2} = 12$$

Out[5]: $(-2)^t + 4t$

In [6]:
```
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 1
```

```python
x[1] = 1
for t in index_set[1:]:
    x[t] = -x[t-1] + 2*x[t-2] + 12
plt.plot(index_set, x)
```

Out[6]: [<matplotlib.lines.Line2D at 0x214dd661df0>]



Example 3

In [7]:
```python
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
eq1 = Eq(yt2 + yt1 - 2*yt, 12)
display(eq1)

from sympy.abc import t,c,k
y = Function("y");


f = y(t+2) + y(t+1) - 2*y(t)- 12;
sol = rsolve(f, y(t), {y(0):4, y(1):5});
sol
```

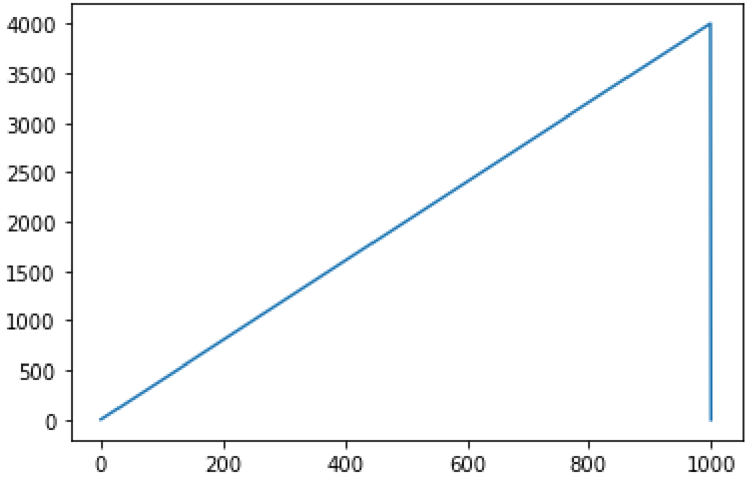
$$-2y_t + y_{t+1} + y_{t+2} = 12$$

Out[7]: $(-2)^t + 4t + 3$

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
N = 1000
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 4
x[1] = 5
for t in index_set[1:N]:
    x[t] = -x[t-1] + 2*x[t-2] + 12

plt.plot(index_set, x)
```

Out[8]: [<matplotlib.lines.Line2D at 0x214dd6cc6a0>]



Example 4

In [9]:
```
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
eq1 = Eq(yt2 + 6*yt1 + 9*yt, 4)
display(eq1)

from sympy.abc import t,c,k
y = Function("y");


f = y(t+2) + 6*y(t+1) + 9*y(t) - 4;
```

```
sol = rsolve(f, y(t), {y(0):1, y(1):1});
sol
```

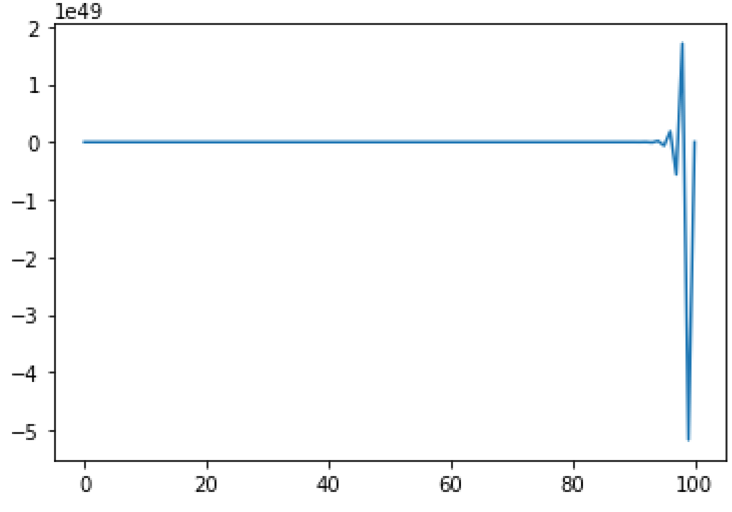
$$9y_t + 6y_{t+1} + y_{t+2} = 4$$

Out[9]: $$(-3)^t \left(\frac{3}{4} - t\right) + \frac{1}{4}$$

In [10]:
```
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 4
x[1] = 5
for t in index_set[1:N]:
    x[t] = -6*x[t-1] - 9*x[t-2] + 4

plt.plot(index_set, x)
```

Out[10]: [<matplotlib.lines.Line2D at 0x214dd7540a0>]



Example 5

In [11]:
```
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
eq1 = Eq(yt2 +  1/4*yt, 5)
display(eq1)
```

```python
from sympy.abc import t,c,k
y = Function("y")
f = y(t+2) + 0*y(t+1) + 1/4*y(t) - 5
sol = rsolve(f, y(t))
sol
```

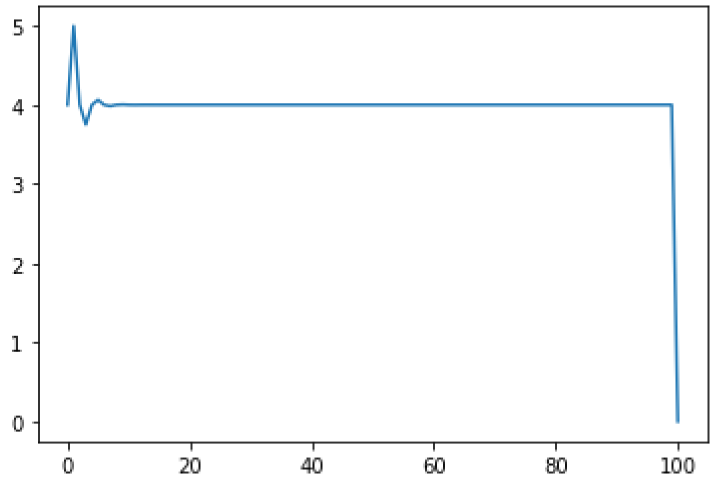
$$0.25y_t + y_{t+2} = 5$$

Out[11]: $$C_0(0.5i)^t + C_1(-0.5i)^t + 4.0$$

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

plt.plot(index_set, x)
```

Out[12]: [<matplotlib.lines.Line2D at 0x214dd7c8970>]



18.2 Samuelson Multiplier-Acceleration Interaction Model

In [13]:
```python
Yt2 = Symbol("Y_t+2")
Yt1 = Symbol('Y_t+1')
```

```
Yt  = Symbol('Y_t')
alpha= Symbol("\\alpha")
gamma= Symbol("\\gamma")
G0 = Symbol('G_0')


eq1 = Eq(Yt2-gamma*(1+alpha)*Yt1 +  alpha*gamma*Yt, G0)
display(eq1)

from sympy.abc import t,c,k
y = Function("y")
f = y(t+2)-gamma*(1+alpha)*y(t+1) + alpha*gamma*y(t) - G0
sol = rsolve(f, y(t))
sol
```

$$Y_t \alpha \gamma - Y_{t+1} \gamma (\alpha + 1) + Y_{t+2} = G_0$$

Out[13]:

$$C_0 \left(\frac{\gamma (\alpha + 1)}{2} - \frac{\sqrt{\gamma (\alpha^2 \gamma + 2\alpha\gamma - 4\alpha + \gamma)}}{2} \right)^t + C_1 \left(\frac{\gamma (\alpha + 1)}{2} + \frac{\sqrt{\gamma (\alpha^2 \gamma + 2\alpha\gamma - 4\alpha + \gamma)}}{2} \right)^t - \frac{G_0}{\gamma - 1}$$

Example 2

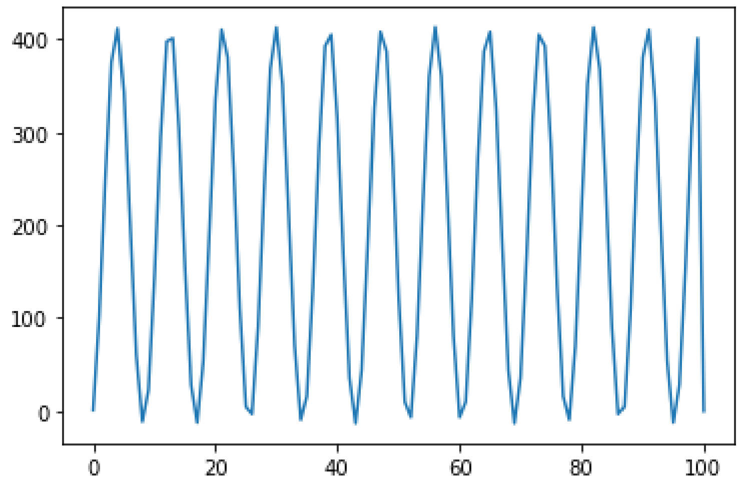
In [15]:

```
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 1
x[1] = 1
for t in index_set[1:N]:
    x[t] =  3/2*x[t-1] - x[t-2] + 100

plt.plot(index_set, x)
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x214dd8be190>]
```



8.3 Inflation and Unemployment in Discrete Time

In [16]:
```
pt2 = Symbol("p_t+2")
pt1 = Symbol('p_t+1')
pt  = Symbol('p_t')
beta= Symbol("\\beta")
k= Symbol("k")
j = Symbol('j')
m = Symbol('m')
g = Symbol("g")


eq1 = Eq( pt2 - ((1+g*j+(1-j)*(1+beta*k)) / (1+beta*k))*pt1 +
         ((1-j*(1-g)))*pt/((1+beta*k)), (j*beta*k*m)/(1+beta*k))
display(eq1)

from sympy import simplify
from sympy.abc import t,c,k
y = Function("y")
f = y(t+2)-((1+g*j+(1-j)*(1+beta*k)))*y(t+1)/(1+beta*k) +(1/(1+beta*k))*((1-j*(1-g)))*y(t) - (j*beta*k*m)/(1+beta*k)
# we just write the 18.24
sol = rsolve(f, y(t))
simplify(sol)
```

$$\frac{p_t\left(-j\left(1-g\right)+1\right)}{\beta k+1}-\frac{p_{t+1}\left(gj+\left(1-j\right)\left(\beta k+1\right)+1\right)}{\beta k+1}+p_{t+2}=\frac{\beta jkm}{\beta k+1}$$

Out[16]:

$$C_0\left(\frac{-\beta jk+\beta k+gj-j-\sqrt{\beta^2j^2k^2-2\beta^2jk^2+\beta^2k^2-2\beta gj^2k-2\beta gjk+2\beta j^2k-2\beta jk+g^2j^2-2gj^2+j^2}+2}{2(\beta k+1)}\right)^t$$
$$+C_1\left(\frac{-\beta jk+\beta k+gj-j+\sqrt{\beta^2j^2k^2-2\beta^2jk^2+\beta^2k^2-2\beta gj^2k-2\beta gjk+2\beta j^2k-2\beta jk+g^2j^2-2gj^2+j^2}+2}{2(\beta k+1)}\right)^t+m$$

18.4 Generalizations to Variable-Term and Higher-Order Equations

In [17]:

```
yt2 = Symbol("y_t+2")
yt1 = Symbol('y_t+1')
yt  = Symbol('y_t')
t  = Symbol('t')
eq1 = Eq(yt2 + yt1 -3*yt, 7**t)
display(eq1)

from sympy.abc import t,c,k
y = Function("y")
f = y(t+2) + y(t+1) -3*y(t) - 7**t
sol = rsolve(f, y(t))
sol
```

$$-3y_t+y_{t+1}+y_{t+2}=7^t$$

Out[17]:

$$\frac{7^t}{53}+C_0\left(-\frac{1}{2}+\frac{\sqrt{13}}{2}\right)^t+C_1\left(-\frac{\sqrt{13}}{2}-\frac{1}{2}\right)^t$$

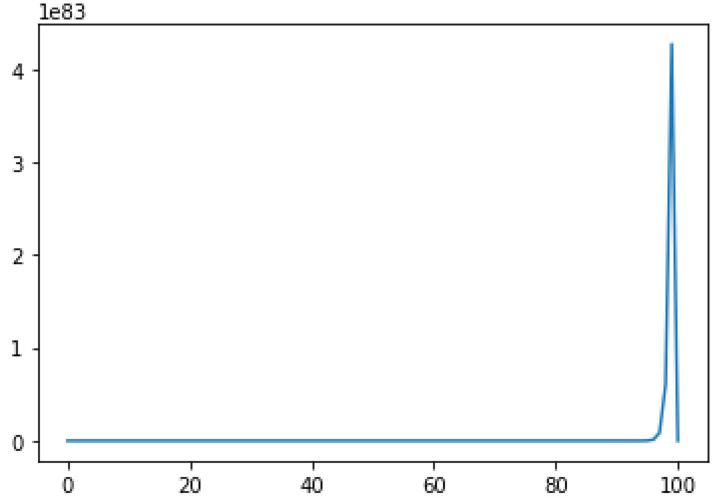
In [18]:

```
N = 100
index_set = range(N+1)
x = np.zeros(len(index_set))
x[0] = 1
x[1] = 1
for t in index_set[1:N]:
    x[t] =  -x[t-1] + 3*x[t-2] + 7**t

plt.plot(index_set, x)
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x214dd99e610>]
```



Furkan zengin