Econ I Module

Juergen Jung

Towson University, 2012

The cake eating problem

Once upon a time there was a little girl who got a cake. The girl decided to eat the cake all alone. But she was not sure when she wanted to eat the cake. First, she thought of eating the whole cake right away. But then, nothing would be left for tomorrow and the day after tomorrow.

Well, on the one hand, eating cake today is better than eating eat it tomorrow. On the other hand, eating too much at the same time might not be the best. She imagined that the first mouthful of cake is a real treat, the second is great, the third is also nice. But the more you eat, the less you enjoy it.

So, she decided to eat only a bit of the cake everyday. Then, she could eat everyday another first mouthful of cake. The girl knew that the cake would be spoiled if she kept it more than nine days. Therefore, she would eat the cake in the first ten days. Yet, how much should she eat everyday?

## Function definitions

The objective function:

```
func1 = function(cv) {
    T = length(cv)
    uv = seq(T)
    for (i in 1:T) {
        beta = 0.9
        uv[i] = beta^(i - 1) * log(cv[i])  # period utilit
    }
    return(-sum(uv))
}
```

# Function definitions The constraint: `r eqn1 = function(cv) { k0 = 100 z1 = sum(cv) - k0 return(c(z1)) }`

# Function definitions

Call the optimizer with some starting values for the consumption

Python

‘‘‘python import math from pylab import *

T = 9 beta = 0.9 kv = zeros(T+1,float) cv = zeros(T+1,float) uv = zeros(T+1,float) kv[0] = 100 # k0 cv[0] = (1.0-beta)/(1.0-beta**(T+1)) * kv[0] # c0 uv[0] = log(cv[0])

for i in range(1,T+1): #print “i=” + str(i) cv[i] = beta * cv[i-1] kv[i] = kv[i-1] - cv[i-1] uv[i] = beta**(i-1)*log(cv[i]) # period utility with discounting

sum(uv) # total utility

print “cv=” +str(cv) print “kv=” +str(kv) cv= [15.35339933 13.8180594 12.43625346 11.19262811 10.0733653 9.06602877 8.15942589 7.3434833 6.60913497 5.94822148] kv= [100. 84.64660067 70.82854128 58.39228782 47.19965971 37.12629441 28.06026564 19.90083975 12.55735645 5.94822148] ‘‘‘

Plotting is achieved via:

Numerical solution in Python

Define functions:

```
def func1(cv):
    T = len(cv)
    uv= zeros(T,float)
    for i in range(T):
        beta = 0.9
        uv[i] = (beta**i) * log(cv[i])  # period utility wi
    return (-sum(uv))

# The constraint
def eqn1(cv):
    k0 = 100
    z1=sum(cv) - k0
    return array([z1])
```