Stochastic Processes and Advanced Mathematical Finance

A Stochastic Process Model of Cash Management

Rating

Mathematically Mature: may contain mathematics beyond calculus with proofs.
Section Starter Question

Suppose that you have a stock of 5 units of a product. It costs you $r$ dollars per unit of product to hold the product for a week. You sell and eliminate from the stock one unit of product per week. What is the total cost of holding the product? Now suppose that the amount of product sold each week is determined by a coin-tossing game (or equivalently a random walk.) How would you calculate the expected cost of holding the product?

---

Key Concepts

1. The **reserve requirement** is a bank regulation that sets the minimum reserves of cash a bank must hold on hand for customer deposits. An important question for the bank is: What is the optimal level of cash for the bank to hold?

2. We model the cash level with a sequence of cycles or games. Each cycle begins with $s$ units of cash on hand and ends with either a replenishment of cash, or a reduction of cash. In between these levels, the cash level is a stochastic process, specifically a coin-tossing game or random walk.

3. By solving a non-homogeneous difference equation we can determine the expected number of visits to an intermediate level in the random process.

4. Using the expected number of visits to a level we can model the expected costs of the reserve requirement as a function of the maximum amount to hold and the starting level after a buy or sell. Then we can minimize the expected costs with calculus to find the optimal values of the maximum amount and the starting level.
Vocabulary

1. The **reserve requirement** is a bank regulation that sets the minimum reserves of cash a bank must hold for customer deposits.

2. The mathematical expression $\delta_{sk}$ is the **Kronecker delta**

$$\delta_{sk} = \begin{cases} 
1 & \text{if } k = s \\
0 & \text{if } k \neq s .
\end{cases}$$

3. If $X$ is a random variable assuming some values including $k$, the **indicator random variable** is

$$1_{\{X=k\}} = \begin{cases} 
1 & X = k \\
0 & X \neq k.
\end{cases}$$

The indicator random variable indicates whether a random variable assumes a value, or is in a set. The expected value of the indicator random variable is the probability of the event.

Mathematical Ideas

**Background**

The **reserve requirement** is a bank regulation that sets the minimum reserves of cash a bank must hold on hand for customer deposits. This is also called the **Federal Reserve requirement** or the **reserve ratio**. These
reserves exist so banks can satisfy cash withdrawal demands. The reserves also help regulate the national money supply. Specifically in 2010 the Federal Reserve regulations require that the first $10.7 million are exempt from reserve requirements. A 3 percent reserve ratio is assessed on net transaction accounts over $10.7 million up to and including $55.2 million. A 10 percent reserve ratio is assessed on net transaction accounts in excess of $55.2 million.

Of course, bank customers are frequently depositing and withdrawing money so the amount of money for the reserve requirement is constantly changing. If customers deposit more money, the cash on hand exceeds the reserve requirement. The bank would like to put the excess cash to work, perhaps by buying Treasury bills. If customers withdraw cash, the available cash can fall below the required amount to cover the reserve requirement so the bank gets more cash, perhaps by selling Treasury bills.

The bank has a dilemma: buying and selling the Treasury bills has a transaction cost, so the bank does not want to buy and sell too often. On the other hand, excess cash could be put to use by loaning it out, and so the bank does not want to have too much cash idle. What is the optimal level of cash that signals a time to sell, and how much should be bought or sold?

**Modeling**

We assume for a simple model that a bank’s cash level fluctuates randomly as a result of many small deposits and withdrawals. We model this by dividing time into successive, equal length periods, each of short duration. The periods might be weekly, the reporting period the Federal Reserve Bank requires for some banks. In each time period, assume the reserve randomly increases or decreases one unit of cash, perhaps measured in units of $100,000, each with probability 1/2. That is, in period $n$, the change in the banks reserves is

$$Y_n = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2. \end{cases}$$

The equal probability assumption simplifies calculations for this model. It is possible to relax the assumption to the case $p \neq q$, but we will not do this here.

Let $T_0 = s$ be the initial cash on hand. Then $T_n = T_0 + \sum_{j=1}^{n} Y_j$ is the total cash on hand at period $n$.

The bank will intervene if the reserve gets too small or too large. Again
for simple modeling, if the reserve level drops to zero, the bank sells assets such as Treasury bonds to replenish the reserve back up to $s$. If the cash level ever increases to $S$, the bank buys Treasury bonds to reduce the reserves to $s$. What we have modeled here is a version of the Gambler’s Ruin, except that when this “game” reaches the “ruin” or “victory” boundaries, 0 or $S$ respectively, the “game” immediately restarts again at $s$.

Now the cash level fluctuates in a sequence of cycles or games. Each cycle begins with $s$ units of cash on hand and ends with either a replenishment of cash, or a reduction of cash. See Figure 1 for a simple example with $s = 2$ and $S = 5$.

**Mean number of visits to a particular state**

Now let $k$ be one of the possible reserve states with $0 < k < S$ and let $W_{sk}$ be the expected number of visits to the level $k$ up to the ending time of the cycle starting from $s$. A formal mathematical expression for this expression is

$$W_{sk} = E \left[ \sum_{j=1}^{N-1} 1_{\{T_j=k\}} \right]$$

where $1_{\{T_j=k\}}$ is the indicator random variable where

$$1_{\{T_j=k\}} = \begin{cases} 1 & T_j = k \\ 0 & T_j \neq k. \end{cases}$$
Note that the inner sum is a random sum, since it depends on the length of the cycle $N$, which is cycle dependent.

Then using first-step analysis $W_{sk}$ satisfies the equations

$$W_{sk} = \delta_{sk} + \frac{1}{2} W_{s-1,k} + \frac{1}{2} W_{s+1,k},$$

with boundary conditions $W_{0k} = W_{Sk} = 0$. The term $\delta_{sk}$ is the Kronecker delta

$$\delta_{sk} = \begin{cases} 1 & \text{if } k = s \\ 0 & \text{if } k \neq s. \end{cases}$$

The explanation of this equation is very similar to the derivation of the equation for the expected duration of the coin-tossing game. The terms $\frac{1}{2} W_{s-1,k} + \frac{1}{2} W_{s+1,k}$ arise from the standard first-step analysis or expectation-by-conditioning argument for $W_{sk}$. The non-homogeneous term in the prior expected duration equation (which is +1) arises because the game will always be at least 1 step longer after the first step. In the current equation, the $\delta_{sk}$ non-homogeneous term arises because the number of visits to level $k$ after the first step will be 1 more if $k = s$ but the number of visits to level $k$ after the first step will be 0 more if $k \neq s$.

For the ruin probabilities, the difference equation was homogeneous, and we only needed to find the general solution. For the expected duration, the difference equation was non-homogeneous with a non-homogeneous term which was the constant 1, making the particular solution reasonably easy to find. Now the non-homogeneous term depends on the independent variable, so solving for the particular solution will be more involved.

First we find the general solution $W^h_{sk}$ to the homogeneous linear difference equation

$$W^h_{sk} = \frac{1}{2} W^h_{s-1,k} + \frac{1}{2} W^h_{s+1,k}.$$ 

This is easy, we already know that it is $W^h_{sk} = A + Bs$.

Then we must find a particular solution $W^p_{sk}$ to the non-homogeneous equation

$$W^p_{sk} = \delta_{sk} + \frac{1}{2} W^p_{s-1,k} + \frac{1}{2} W^p_{s+1,k}.$$ 

For purposes of guessing a plausible particular solution, temporarily re-write the equation as

$$-2\delta_{sk} = 2W^p_{s-1,k} - 2W^p_{sk} + W^p_{s+1,k}.$$
The expression on the right is a centered second difference. For the prior expected duration equation, we looked for a particular solution with a constant centered second difference. Based on our experience with functions it made sense to guess a particular solution of the form \( C + Ds + Es^2 \) and then substitute to find the coefficients. Here we seek a function whose centered second difference is 0 except at \( k \) where the second difference jumps to 1. This suggests the particular solution is piecewise linear, say

\[
W_{sk}^p = \begin{cases} 
C + Ds & \text{if } s \leq k \\
E + Fs & \text{if } s > k.
\end{cases}
\]

In the exercises, we verify that the coefficients of the function are

\[
W_{sk}^p = \begin{cases} 
0 & \text{if } s < k \\
2(k - s) & \text{if } s \geq k.
\end{cases}
\]

We can write this as \( W_{sk}^p = -2 \max(s - k, 0) \).

Then solving for the boundary conditions, the full solution is

\[
W_{sk} = 2 \left[ s(1 - k/S) - \max(s - k, 0) \right].
\]

**Expected Duration and Expected Total Cash in a Cycle**

Consider the **first passage time** \( N \) when the reserves first reach 0 or \( S \), so that cycle ends and the bank intervenes to change the cash reserves. The value of \( N \) is a random variable, it depends on the sample path. We are first interested in \( D_s = \mathbb{E}[N] \), the expected duration of a cycle. From the previous section we already know \( D_s = s(S - s) \).

Next, we are interested in the mean cost of holding cash on hand during a cycle \( i \), starting from amount \( s \). Call this mean \( W_s \). Let \( r \) be the cost per unit of cash, per unit of time. We then obtain the cost by weighting \( W_{sk} \), the mean number of times the cash is at number of units \( k \) starting from \( s \), multiplying by \( k \), multiplying by the factor \( r \) and summing over all the
available amounts of cash:

\[ W_s = \sum_{k=1}^{s-1} rkW_{sk} \]

\[ = 2 \left[ \frac{s}{S} \sum_{k=1}^{s-1} rk(S - k) - \sum_{k=1}^{s-1} rk(s - k) \right] \]

\[ = 2 \left[ \frac{s}{S} \left( rs(S - 1)(S + 1) \right) - \frac{rs(s - 1)(s + 1)}{6} \right] \]

\[ = r \frac{s}{3} \left[ S^2 - s^2 \right]. \]

These results are interesting and useful in their own right as estimates of the length of a cycle and the expected cost of cash on hand during a cycle. Now we use these results to evaluate the long run behavior of the cycles. Upon resetting the cash at hand to \( s \) when the amount of cash reaches 0 or \( S \), the cycles are independent of each of the other cycles because of the assumption of independence of each step. Let \( K \) be the fixed cost of the buying or selling of the treasury bonds to start the cycle, let \( N_i \) be the random length of the cycle \( i \), and let \( R_i \) be the total opportunity cost of holding cash on hand during cycle \( i \). Then the cost over \( n \) cycles is \( nK + R_1 + \cdots + R_n \). Divide by \( n \) to find the average cost

\[ \text{Expected total cost in cycle } i = K + \mathbb{E}[R_i], \]

but we have another expression for the expectation \( \mathbb{E}[R_i] \),

\[ \text{Expected opportunity cost} = \mathbb{E}[R_i] = r \frac{s}{3} \left[ S^2 - s^2 \right]. \]

Likewise the total length of \( n \) cycles is \( N_1 + \cdots + N_n \). Divide by \( n \) to find the average length,

\[ \text{Expected length} = \frac{N_1 + \cdots + N_n}{n} = s(S - s). \]

These expected values allow us to calculate the average costs

\[ \text{Long run average cost, dollars per week} = \frac{K + \mathbb{E}[R_i]}{\mathbb{E}[N_i]}. \]
Then $\mathbb{E}[R_0] = rW_s$ and $\mathbb{E}[N_0] = s(S - s)$. Therefore

$$\text{Long run average cost, dollars per week} = \frac{K + (1/3)rs(S^2 - s^2)}{s(S - s)}.$$ 

Simplify the analysis by setting $x = s/S$ so that the expression of interest is

$$\text{Long run average cost} = \frac{K + (1/3)rS^3x(1 - x^2)}{S^2x(1 - x)}.$$ 

Remark. Aside from being a good thing to non-dimensionalize the model as much as possible, it also appears that optimizing the original long run cost average in the original variables $S$ and $s$ is messy and difficult. This of course would not be known until you had tried it. However, knowing the optimization is difficult in variables $s$ and $S$ additionally motivates making the transformation to the non-dimensional ratio $x = s/S$.

Now we have a function in two variables that we wish to optimize. Take the partial derivatives with respect to $x$ and $S$ and set them equal to 0, then solve, to find the critical points. The results are that

$$x_{\text{opt}} = \frac{1}{3},$$

$$S_{\text{opt}} = 3\left(\frac{3K}{4r}\right)^{\frac{1}{3}}.$$ 

That is, the optimal value of the maximum amount of cash to keep varies as the cube root of the cost ratios, and the reset amount of cash is $1/3$ of that amount.

**Criticism of the model**

The first test of the model would be to look at the amounts $S$ and $s$ for well-managed banks and determine if the banks are using optimal values. That is, one could do a statistical survey of well-managed banks and determine if the values of $S$ vary as the cube root of the cost ratio, and if the restart value is $1/3$ of that amount. Of course, this assumes that the model is valid and that banks are following the predictions of the model, either consciously or not.
This model is too simple and could be modified in a number of ways. One change might be to change the reserve requirements to vary with the level of deposits, just as the 2010 Federal Reserve requirements vary. Adding additional reserve requirement levels to the current model adds a level of complexity, but does not substantially change the level of mathematics involved.

The most important change would be to allow the changes in deposits to have a continuous distribution instead of jumping up or down by one unit in each time interval. Modification to continuous time would make the model more realistic instead of changing the cash at discrete time intervals. The assumption of statistical independence from time step to time step is questionable, and so could also be relaxed. All these changes require deeper analysis and more sophisticated stochastic processes.

Sources

This section is adapted from: Section 6.1.3 and 6.2, pages 157-164 in *An Introduction to Stochastic Modeling*, [1].

Algorithms, Scripts, Simulations

Set the top boundary state value and the start and reset state value. Set the probability of a transition of an interior state to the next higher state. Set the number of steps in the Markov chain. Create the Markov chain transition probability matrix. For this Markov chain, the transition probability matrix is tridiagonal, with 0 on the main diagonal, $p$ on the upper diagonal and $1-p$ on the lower diagonal. For the boundary states, the transition probability is 1 to the start or reset state.

Initialize the vector holding the number of visits to each state, the number of cycles from start to reset, and the length of each cycle. For each transition, choose a random value $u$ chosen from a uniform distribution on $(0, 1)$. Starting from an initialized cumulative probability of 0, compute the cumulative probability of the current state’s transition probability distribution.
Compare the cumulative probability to $u$ until the cumulative probability exceeds $u$. This is the new state. Update the number of visits to each state, the number of cycles from start to reset, and the length of each cycle and at the end of a cycle compute the average number of visit and the average length of the cycle. Print the average number of visits, and compare to the theoretical number of visits, Or for cash management, compute the actual cash management cost and the theoretical cash management value.

Markov Chain Algorithm

R script for markov chain.

```r
n <- 10 # Top boundary, number of states 0..n is n+1
s <- 5 # Start and Reset state number 1 <= s <= n-1
p <- 1/2
steps <- 1000

diag.num <- outer(seq(1,n+1),seq(1,n+1), "-"
# diag.num is a matrix whose ith lower diagonal equals i,
# opposite for upper diagonal
T <- mat.or.vec(n+1,n+1)
# mat.or.vec creates an nr by nc zero matrix if nc is
# greater than 1
# Also remember that matrices in R are 1-based, so need n +1 states, 0..n
T[diag.num == -1] <- p
T[diag.num == 1] <- 1-p
T[1,2] <- 0; T[1,s+1] <- 1;
T[n+1,n] <- 0; T[n+1,s+1] <- 1;

# vector to hold the count of visits to each state during a cycle
count <- mat.or.vec(1, n+1)
# Initialize the number of cycles
numberCycles <- 0
# Initialize the length of the cycle
cycleLength <- 0
# Start in the state s
state = s+1

# Make steps through the markov chain
for (i in 1:steps)
{
   x = 0;
}
```

11
```octave
n = 10;  # Top boundary, number of states 0..n is n+1
s = 5;   # Start and Reset state number 1 <= s <= n-1
p = 1/2;
```

```octave
u = runif(1, 0, 1);

newState = state;
for (j in 1:ncol(T))
{
    x = x + T[state, j];
    if (x >= u)
    {
        newState = j;
        break;
    }
}
## newState <- sample(1:ncol(T), 1, prob=T[state,])
state = newState;
count[state] <- count[state] + 1
cycleLength <- cycleLength + 1
if (state == n+1 || state == 1) {
    numberCycles <- numberCycles + 1
    avgVisits <- count/numberCycles
    avgCycleLength <- i/numberCycles
}
Wsk <- avgVisits
theoreticalWsk <- 2*( s*(1-(0:n)/n) - pmax ( s - (0:n),0) );
cat(sprintf("Average number of visits to each state in a
cycle: \n "))
cat(Wsk)
cat("\n\n")
cat(sprintf("Theoretical number of visits to each state
in a cycle: \n "))
cat(theoreticalWsk)
cat("\n")
```

**Octave**  
Octave script for markov chain
steps = 1000;
rate = 1.0;
K = 2;  # Charge or cost to reset the Markov chain
T = diag( p*ones(1,n), 1) + diag( (1-p)*ones(1,n), -1);
T(1,2) = 0; T(1,s+1) = 1;
T(n+1,n) = 0; T(n+1,s+1) = 1;

# vector to hold the count of visits to each state during a cycle
count = zeros(1, n+1);
# Initialize the cycle length
numberCycles = 0;
# Start in the state s
state = s+1;

# Make steps through the markov chain
for i = 1:steps
    x = 0;
    u = rand(1, 1);

    newState = state;
    for j = 1:n+1
        x = x + T(state, j);
        if (x >= u)
            newState = j;
            break;
        endif
    endfor
    ## newState = sample(1:ncol(T), 1, prob=T[state,])
    state = newState;
    count(state) = count(state) + 1;
    if (state == n+1 || state == 1)
        numberCycles = numberCycles + 1;
        avgVisits = count/numberCycles;
        avgCycleLength = i/numberCycles;
    endif
endfor

disp("Average number of visits to each state in a cycle:"
     )
Wsk = avgVisits
disp("Theoretical number of visits to each state in a cycle")
theoreticalWsk = 2*( s*(1-(0:n)/n) - max( s - (0:n), zeros(1,n+1) ) )

Perl

Perl PDL script for markov chain

```perl
use PDL::NiceSlice;

$n = 10; # Top boundary, number of states 0..n is n+1
$s = 5; # Start and Reset state number 1 <= s <= n-1
$p = 1 / 2;
$steps = 1000;

$T = zeroes( $n + 1, $n + 1 );
$T ( $s, 0 ) .= 1; # note order of indices, note concat operation
$T ( $s, $n ) .= 1;
for ( $i = 1; $i <= $n - 1; $i++ ) {
    $T ( $i + 1, $i ) .= $p;
    $T ( $i - 1, $i ) .= 1 - $p;
}

# vector to hold the count of visits to each state during a cycle
$count = zeroes( $n + 1 );

# Initialize the number of cycles
$numberCycles = 0;

# Start in the state s
$state = $s;

# Make steps through the markov chain
for ( $i = 1; $i <= $steps; $i++ ) {
    $x = 0;
    $u = rand(1);
    for ( $j = 0; $j <= $n; $j++ ) {
        $x = $x + $T ( $j, $state );
    }
    if ( $x >=$u ) {
        $state = $j;
```
last;
}

$count ($state)++;
if ( $state == $n || $state == 0 ) {
    $numberCycles++;
    $avgVisits = $count / $numberCycles;
    $avgCycleLength = $i / $numberCycles;
}

$Wsk = $avgVisits;
$theoreticalWsk = 2 * ( $s * ( 1 - sequence( $n + 1 ) / $n ) - maximum( pdl( [ $s - sequence( $n + 1 ), zeroes($n) ] ) -> xchg( 0, 1 ) ));

print "Average number of visits to each state in a cycle :
", $Wsk, "\n";
print "Theoretical number of visits to each state in a
cycle: \n", $theoreticalWsk, "\n";

SciPy

```python
import scipy

n = 10;  # Top boundary, number of states 0..n is n+1
s = 5;   # Start and Reset state number 1 <= s <= n-1
p = 0.5;
steps = 1000;

T = scipy.diag( p*scipy.ones( n ), 1) + scipy.diag( (1-p)*scipy.ones( n ), -1);
T[0,1] = 0; T[0,s] = 1;
T[n,n-1] = 0; T[n,s] = 1;

# vector to hold the count of visits to each state during a cycle
count = scipy.zeros(n+1, float);
```
# Initialize the cycle length
numberCycles = 0;

# Start in the state s
state = s;

# Make steps through the markov chain
for i in range(1, steps+1):
    x = 0;
    u = scipy.random.random(1);
    newState = state;
    for j in range(n+1):
        x = x + T[state, j];
        if (x >= u):
            newState = j;
            break;
    ## newState = scipy.random.choice(1, 1, prob=T[state ,])
    state = newState;
    count[state] = count[state] + 1;
    if (state == n or state == 0):
        numberCycles = numberCycles + 1;
        avgVisits = count/numberCycles;
        avgCycleLength = i/numberCycles;

Wsk = avgVisits
theoreticalWsk = 2*( s*(1-scipy.arange(0,n+1,dtype=float)/n) - scipy.maximum(s-scipy.arange(0,n+1), scipy.zeros(n+1, int) ) )
print "Average number of visits to each state in a cycle: \n", Wsk
print "Theoretical number of visits to each state in a cycle: \n", theoreticalWsk

**Cash Management Algorithm**

**GeoGebra** GeoGebra script for cash management

**R** R script for cash management.

```R
n <- 10  # Top boundary, number of states 0..n is n+1
s <- 5   # Start and Reset state number 1 <= s <= n-1
p <- 1/2
```
steps <- 1000
rate <- 1.0
K <- 2  # Charge or cost to reset the Markov chain

diag.num <- outer(seq(1,n+1),seq(1,n+1), "-")
# diag.num is a matrix whose ith lower diagonal equals i, opposite for upper diagonal
T <- mat.or.vec(n+1,n+1)
# mat.or.vec creates an nr by nc zero matrix if nc is greater than 1
# Also remember that matrices in R are 1-based, so need n +1 states, 0..n
T[diag.num == -1] <- p
T[diag.num == 1] <- 1-p
T[1,2] <- 0; T[1,s+1] <- 1;
T[n+1,n] <- 0; T[n+1,s+1] <- 1;

# vector to hold the count of visits to each state during a cycle
count <- mat.or.vec(1, n+1)
# Initialize the number of cycles
numberCycles <- 0
# Initialize the total cost of the cashmanagement
totalCost <- 0
# Start in the state s
state = s+1

# Make steps through the markov chain
for (i in 1:steps)
{
  x = 0;
  u = runif(1, 0, 1);
  newState = state;
  for (j in 1:ncol(T))
  {
    x = x + T[state, j];
    if (x >= u)
    {
      newState = j;
      break;
    }
  }
  ## newState <- sample(1:ncol(T), 1, prob=T[state,])
  state = newState;
Octave script for cash management

```octave
n = 10;  # Top boundary, number of states 0..n is n+1
s = 5;  # Start and Reset state number 1 <= s <= n-1
p = 1/2;
steps = 1000;
rate = 1.0;
K = 2;  # Charge or cost to reset the Markov chain

T = diag( p*ones(1,n), 1) + diag( (1-p)*ones(1,n), -1);
T(1,2) = 0; T(1,s+1) = 1;
T(n+1,n) = 0; T(n+1,s+1) = 1;

# vector to hold the count of visits to each state during a cycle
count = zeros(1, n+1);
# Initialize the number of cycles
numberCycles = 0;
# Initialize the total cost of the cash management
totalCost = 0;
# Start in the state s
state = s+1;
```
# Make steps through the markov chain

for i = 1:steps
    x = 0;
    u = rand(1, 1);

    newState = state;
    for j = 1:n+1
        x = x + T(state, j);
        if (x >= u)
            newState = j;
            break;
    endif
endfor

## newState = sample(1:ncol(T), 1, prob=T[state,])
state = newState;
count(state) = count(state) + 1;

if (state == n+1 || state == 1)
    numberCycles = numberCycles + 1;
    totalCost = K + totalCost;
else
    totalCost = rate*(state-1) + totalCost;
endif
endfor

disp("Average cost:")
avgCost = totalCost/steps
disp("Theoretical average cost:")
theoreticalAvgCost = ( K + (1/3)*(rate*s*(n^2 - s^2) ) )/(s*(n-s) )

Perl

use PDL::NiceSlice;

$n   = 10;    # Top boundary, number of states 0..n
    is n+1
$s   = 5;     # Start and Reset state number 1 <= s <= n-1
$p   = 1 / 2;
$steps = 1000;
$rate = 1.0;

$K = 2;  # Charge or cost to reset the Markov chain

$T = zeroes( $n + 1, $n + 1 );

$T ( $s, 0 ) .= 1;  # note order of indices, note concat operation
$T ( $s, $n ) .= 1;

for ( $i = 1; $i <= $n - 1; $i++ ) {
    $T ( $i + 1, $i ) .= $p;
    $T ( $i - 1, $i ) .= 1 - $p;
}

# vector to hold the count of visits to each state during a cycle
$count = zeroes( $n + 1 );

# Initialize the number of cycles
$numberCycles = 0;

# Initialize the total cost of the cashmanagement
$totalCost = 0.0;

# Start in the state s
$state = $s;

# Make steps through the markov chain
for ( $i = 1; $i <= $steps; $i++ ) {
    $x = 0;
    $u = rand(1);
    for ( $j = 0; $j <= $n; $j++ ) {
        $x = $x + $T ( $j, $state );
        if ( $x >= $u ) {
            $state = $j;
            last;
        }
    }
    $count ($state)++;
    if ( $state == $n || $state == 0 ) {
        $numberCycles++;
        $totalCost = $K + $totalCost;
    } else {
        $totalCost = $rate * $state + $totalCost;
    }
}
$avgCost = $totalCost / $steps;

$theoreticalAvgCost =
( $K + ( 1. / 3. ) * ( $rate * $s * ( $n**2 - $s**2 ) ) )
/ ( $s * ( $n - $s ) );

print "Average Cost: ", $avgCost,
"\n";
print "Theoretical Average Cost: ", $theoreticalAvgCost,
"\n";

SciPy

```python
import scipy

n = 10;  # Top boundary, number of states 0..n is n+1
s = 5;   # Start and Reset state number 1 <= s <= n-1
p = 0.5;
steps = 1000;
rate = 1.0
K = 2.0

T = scipy.diag( p*scipy.ones( n ), 1) + scipy.diag( (1-p) *scipy.ones( n ), -1);
# square brakets okay, assignment not
T[0,1] = 0; T[0,s] = 1;
T[n,n-1] = 0; T[n,s] = 1;

# vector to hold the count of visits to each state during a cycle
count = scipy.zeros(n+1, float);
# Initialize the cyclelength
numberCycles = 0;
# Initialize the total cost of the cashmanagement
totalCost = 0.0
# Start in the state s
state = s;

# Make steps through the markov chain
for i in range(1, steps+1):
```

21
\begin{verbatim}
x = 0;
u = scipy.random.random(1);
newState = state;
for j in range(n+1):
x = x + T[state, j];
if (x >= u):
    newState = j;
    break;
## newState = scipy.random.choice(1, 1, prob=T[state ,])
state = newState;
count[state] = count[state] + 1;
if (state == n or state == 0):
    numberCycles = numberCycles + 1;
totalCost = K + totalCost
else:
totalCost = rate*state + totalCost
avgCost = totalCost/steps
theoreticalAvgCost = ( K + (1./3.)*(rate*s*(n**2 - s**2) ) )/( s*(n-s) )
print "Average cost:\n", avgCost
print "Theoretical average cost: \n", theoreticalAvgCost
\end{verbatim}

\textbf{Problems to Work for Understanding}

1. Find a particular solution $W^p_{sk}$ to the non-homogeneous equation

\[ W^p_{sk} = \delta_{sk} + \frac{1}{2} W^p_{s-1,k} + \frac{1}{2} W^p_{s+1,k}, \]

using the trial function

\[ W^p_{sk} = \begin{cases} 
C + Ds & \text{if } s \leq k \\
E + Fs & \text{if } s > k.
\end{cases} \]
2. Show that

\[ W_s = \sum_{k=1}^{s-1} kW_{sk} \]

\[ = 2 \left[ \frac{s}{S} \sum_{k=1}^{s-1} k(S - k) - \sum_{k=1}^{s-1} k(s - k) \right] \]

\[ = 2 \left[ \frac{s}{S} \left[ \frac{S(S - 1)(S + 1)}{6} \right] - \frac{s(s - 1)(s + 1)}{6} \right] \]

\[ = \frac{s}{3} [S^2 - s^2] \]

You will need formulas for \( \sum_{k=1}^{N} k \) and \( \sum_{k=1}^{N} k^2 \) or alternatively for \( \sum_{k=1}^{N} k(M - k) \). These are easily found or derived.

3. (a) For the long run average cost

\[ C = \frac{K + (1/3)rS^3x(1 - x^2)}{S^2x(S - x)}. \]

find \( \frac{\partial C}{\partial x} \).

(b) For the long run average cost

\[ C = \frac{K + (1/3)rS^3x(1 - x^2)}{S^2x(1 - x)}. \]

find \( \frac{\partial C}{\partial S} \).

(c) Find the optimum values of \( x \) and \( S \).

---

Reading Suggestion:

References

Outside Readings and Links: