Appendix 1

The DM owns an asset with liquidation value *L* which earns interest at the rate of return *rr* until the forced liquidation date in period *T*. We start in period 1 with a cash flow x_1. Thereafter the cash flow follows a binomial random walk: given *x* in period *t* the cash flow in period *t+1* is either *x+h* (with probability *p*) or *x-h* (with probability *1-p*).

I start with **Objective Function 1**: the maximisation of the expectation of the sum of the utilities of the payoffs (cash flow plus liquidation when it occurs).

There are various *nodes* that the DM may reach. In period 1 there is just 1; in period 2 there are 2;...; in period *t,* there are *t* of them;…, in period *T* there are *T* of them. The total number of such nodes is *1+2+…+T = T(T+1)/2.* We could refer to these nodes with a *pair* of numbers *(t,j)* where *j* goes from 1 to *t* in period *t.* A better way is to define *k nodes* which go sequentially from 1 to *T(T+1)/2.* In period 1, *k* is just 1; in period 2, *k* is 2 and 3; in period *t, k* goes from *(t-1)t/2+1* to *t(t+1)/2*; in period *T, k* goes from *(T-1)T/2+1* to *T(T+1)/2.* I call the total number of *k* nodes *totk*. This is equal to *T(T+1)/2.*

At each *k* node there is an associated cash flow. Letting x_k denote the cash flow at the node *k*, we have from the binomial process we have the following Matlab code:

```
for t=2:1:T                     % going through the periods
    for j=1:1:t                 % for each period going through the j nodes
        k=(t-1)*t/2+j;          % calculating the corresponding k node
        x(k)=x(1)+(t-2*j+1)*h;  % calculating the cash flow at that k node
    end
end
```

Now let us find the solution for Objective Function 1, where the objective is the maximisation of the expected value of the sum of the utilities. I use the following notation. d_k is the optimal decision at node *k*. EV_k is the expected value of the objective function *as viewed from node k*. At this node the previous elements $u(x_1) + u(x_2) + \dots$ are given and known and therefore do not enter the objective function.

Denote by lqv_k the liquidation value of liquidating at that node, and by ctv_k the continuation (expected) value at that node.

From node *k* the DM either moves Up or Down. We need to know to which *k* nodes these moves take us. From the tree (see Figure 1) it can be seen that if at node *k* in period *t* moving Up takes the DM to node *k+t* in period *t+1,* while moving Down takes the DM to node *k+t+1* in period *t+1*.

We work backwards starting in period *T*. Here there are no decisions to take and we have simply:
for *k* between *(T-1)T/2+1* and *T(T+1)/2* that

(1) $EV_k = u(x_k+L)$ This is for the period *T* nodes.

Now work backwards. Here I take the general case of *k<=(T-1)T/2* (that is in periods 1 through *T-1*). I first write the solution in equations and then transfer it into Matlab code. The backward induction starts in period *T-1* and then works backwards to period 1. In period *t* (Note that in period *t* the index *k* takes values from *(t-1)t/2+1* to *t(t+1)/2* inclusive) the relevant equations are:

(2) $ctv_k = u(x_k) + pEV_{k+t} + (1-p)EV_{k+t+1}$
(3) $lqv_k = u(x_k + Lrr^{T-t})$
(4) $d_k = 1$ if $ctv_k \geq lqv_k$; *0* otherwise (I am assuming a DM who is indifferent continues).

(5) $EV_k = \max[ctv_k, lqv_k]$

The Matlab code follows (note I use here a generic utility function; in the code we distinguish between CRRA and CARA).

```
for t=T-1:-1:1                                % for each period working back
      for k=1+t*(t-1)/2:1:t*(t+1)/2           $ for the k nodes in that period
            ctv(k)=u(x(k))+p*EV(k+t)+(1-p)*EV(k+t+1); % continuation value
            lqv(k)=u(L*(rr^(T-t))+x(k));               % liquidation value
            if lqv(k)<=ctv(k)                 % if continuing is better
                  EV(k)=ctv(k);               % the continuation value is EV
                  d(k)=1;                     % decision is to continue
            end
            if lqv(k)>ctv(k)                  % if liquidating is better
                  EV(k)=lqv(k);               % the liquidation value is EV
                  d(k)=0;                     % the decision is to liquidate
            end
      end
end
```

Now let me turn to **Objective Function 2**, where the objective function is the maximisation of the expected utility of the sum of payoffs. This means that the optimal decision at any k-node depends not only on that node but also the accumulated cash flows at that node. Note crucially that knowing one is at a particular k-node is not sufficient to know the accumulated cash flow at that node; this latter depends upon the *route* by which the DM has reached that node. For example consider $k=5$ in $t=3$. This node could have been reached by going Up from period 1 to 2 and then Down from period 2 to 3; or it could have been reached by going Down from period 1 to 2 and then Up from period 2 to 3. In the former case the accumulated cash flow would be $x_1 + (x_1+h) + x_1 = 3x_1+h$; in the latter case the accumulated cash flow would be $x_1 + (x_1-h) + x_1 = 3x_1-h$. In order to deal with this, I need to introduce what I call l-nodes, indicating not only which k-node the DM is at, but also the accumulated cash flow he or she has. I should note that two different l-nodes do not necessarily have different accumulated cash flows.

How many l-nodes are there? It can be seen from Figure 1 that in period t there are a total of 2^{t-1} l-nodes, half of them reached by going Up from the 2^{t-2} l-nodes in period $t-1$ and half of them reached by going Down from the 2^{t-2} l-nodes in period $t-1$. So the total number of l-nodes in a tree of length T is $1 + 2 + 2^2 + 2^3 + 2^4 + \ldots + 2^{T-1} = 2^T-1$. We need to calculate the optimal decisions at all of these with the exception of the 2^{T-1} nodes in period T where the only decision is to stop. So we have (where the subscript now is the l-node):

(6) $d_l = 0$ if $2^{T-2} + 1 \le l \le 2^{T-1}$.

Also we have (in the final period if it is reached)

(7) $EV_l = u(X_l + L)$ if $2^{T-2} + 1 \le l \le 2^{T-1}$ where X_l denotes the accumulated cash flow at node l, and EV_l denotes the value of the objective function at node l.

Now the optimisation procedure is straightforward. We already have the (default) decisions in the final period and the Expected Value of the objective function at each of the final period l-nodes. So we can write (recall that the vector *upl(l)* tells us to which l-node a movement Up from node l takes the DM, and *dnl(l)* tells us to which l-node a movement Down takes the DM from node l):

For all the other l-nodes in periods $t<T$ we have:

(8) $ctv_l = pEV_{upl(l)} + (1-p)EV_{dnl(l)}$

(9) $lqv_l = u(X_l + Lrr^{T-t})$

(10) $d_l = 1$ if $ctv_l \geq lqv_l$; 0 otherwise (I am assuming a DM who is indifferent continues).

(11) $EV_l = \max[ctv_l, lqv_l]$

Note that in these expressions the value of t is that corresponding to the period in which that l-node is in.

Let me number the l nodes so that we have the following. I am using the numbering in the Matlap program OptStop4 in the appropriate directory.

t	*k*	*e(k) the number of l nodes in the k node*	*l*	up *l* node	in *k* node	down *l*-node	in *k* node
1	1	1	1	2	2	3	3
2	2	1	2	4	4	6	4
	3	1	3	5	5	7	6
3	4	1	4	8	7	11	8
	5	2	5	9	8	13	9
			6	10	8	14	9
	6	1	7	12	9	15	10
4	7	1	8	16	11	20	12
	8	3	9	17	12	24	13
			10	18	12	25	13
			11	19	12	26	13
	9	3	12	21	13	28	14
			13	22	13	29	14
			14	23	13	30	14
	10	1	15	27	14	31	15
5	11	1	16	32	16	37	17
	12	4	17	33	17	44	18
			18	34	17	45	18
			19	35	17	46	18
			20	36	17	47	18
	13	6	21	38	18	52	19
			22	39	18	53	19
			23	40	18	54	19
			24	41	18	55	19
			25	42	18	56	19
			26	43	18	57	19
	14	4	27	48	19	59	20
			28	49	19	60	20
			29	50	19	61	20
			30	51	19	62	20
	15	1	31	58	20	63	21

So the implied tree and the *j, k* and *l* nodes are as follows.

1 (1)	2 (2 to 3)	3 (4 to 7)	4 (8 to 15)	5 (16 to 31)	6 (32 to 63)
					1; 16; *32*
				1; 11; *16*	
			1; 7; *8*		2; 17; *33, 34, 35, 36,* 37
		1; 4; *4*		2; 12; *17, 18, 19,*20	
	1; 2; *2*		2; 8; *9, 10*, 11		3; 18; *38, 39, 40, 41, 42, 43,* 44, 45, 46, 47
1; 1; 1		2; 5; *5*, 6		3; 13; *21, 22, 23,* 24, 25, 26	
	2; 3; 3		3; 9; *12*, 13, 14		4; 19; *48, 49, 50, 51,* 52,53,54, 55, 56, 57
		3; 6; 7		4; 14; *27,* 28, 29, 30	

			4; 10; 15		5; 20; *58*, 59, 60, 61, 62
				5; 15; 31	
					6; 21; 63

The numbers in the top row are the *t*-values.
At each node, the first number is what I call the *j*-value; the second number is the *k*-node; and all the other numbers are the *l*-nodes. Of these other numbers, the ones in normal font are the *l*-nodes reached by coming DOWN from the previous period, and those in *italics* those reached by coming UP.

So if we number the *l* nodes this way, it is nice and simple: *l* goes up to *2l* and goes down to *2l+1*, for all *l* from 1 to $2^{T-2}-1$ (up to the penultimate period)

Moreover the accumulated cash flow at node *l* is the cash flow in the associated *k* node plus the accumulated cash flow in the *k* node from where it came.

(12) $X(l) = X(l/2)+x(k)$ if *l* is even
(13) $X(l) = X((l-1)/2)+x(k)$ if *l* is odd
where *k* is the *k* node in which *l* is.
We should do this for all *l* from 2 to 2^{T-1}.

Now how to find the *k* node in which a particular *l* value is. The following Matlab code appears to work (it is in *testclk.m*). Note that there are $(t-1)!/[(j-1)!(t-j)!]$ *l* nodes in node (t,j). This expression is calculated using *nchoosek* in Matlab.

```
l=0;
k=0;
clk(1)=1;
for t=2:1:T
      for j=1:1:t
            k=k+1;
            nl=nchoosek(t-1,j-1);
                  for ll=1:1:nl
                        l=l+1;
                        clk(l)=k;
                  end
      end
end
```

We also need to know (see above for the liquidation values) the *t* node corresponding to a particular *k* node. Here is the Matlab code the vector *ckt*:

```
%now we need to find the t value corresponding to any k value
k=0;
for t=1:1:T
    for j=1:1:t
        k=k+1;
        ckt(k)=t;
    end
end

%this does the important stuff
for l=2^(T-1):1:2^T-1;  %these are the period T nodes
    if rt==1
       EV(l)=crra(L+X(l),r);
    end
```

```
    if rt==2
       EV(l)=cara(L+X(l),r);
    end
end

%this is the important recursion for the other periods going backwards
% this is for a generic utility function
for t=T-1:-1:1
    for l=2^(t-1):1:2^t-1
        kk=clk(l);
        tt=ckt(kk);
        ctv(l)=p*EV(2*l)+(1-p)*EV(2*l+1);
        lqv(l)=u(L*(rr^(T-tt))+X(l));
        if lqv(l)<=ctv(l)
            EV(l)=ctv(l);
            d(l)=1;
        end
        if lqv(l)>ctv(l)
            EV(l)=lqv(l);
            d(l)=0;
        end
    end
end
```

Finally let me show the decisions of a DM with a **rolling strategy**. Here we assume risk-neutrality.

We need to start with the fully-optimal strategy – backwardly inducting from the end. Let us denote the *Expected Value* to the decision-maker of fully optimising if he or she is at node *k* by $EV_{T,k}$ (the first argument indicating the horizon used by the decision-maker and the second the node). Let us denote by $D_{T,k}$ the optimal decision, taking the value 1 for continuing and the value 0 for liquidating. We work backwards. I am now making the notation consistent with the Matlab code.
In *T* we have (ignoring the accumulated cash flows which are given and the decision-maker will get anyhow):

(14) $D_{T,k} = 0$ for *k* from $1+(T-1)T/2$ to $T(T+1)/2$

(15) $EV_{T,k} = x_k + L$ for *k* from $1+(T-1)T/2$ to $T(T+1)/2$

Now we work *backwards*, from $t=T-1$ to $t=1$, using the following recursion. Note that if the DM is at node *k* in period *t,* then going up arrives at node $k+t$ in period $t+1$, and going down arrives at node $k+t+1$ in period $t+1$.

(16) $D_{T,k} = 0$ if $x_k + Lr^{(T-t)} > pEV_{T,k+t} + (1-p)EV_{T,k+t+1}$
$= 1$ if $x_k + Lr^{(T-t)} \leq pEV_{T,k+t} + (1-p) EV_{T,k+t+1}$

(17) $EV_{T,k} = max[x_k + Lr^{(T-t)}, x_{tj} + pEV_{T,k+t} + (1-p) EV_{T,k+t+1}]$

So we have the optimal decision at each cash flow node.

Now let us consider someone who has a rolling strategy with an horizon of *S* periods – so in period t works *as if* he or she *has* to liquidate in period *t+S* or in period *T* whichever is the sooner (the true liquidation date is *T*). Let us use $d_{S,T,k}$ to denote the decision of such a decision-maker at node *k*, the first argument indicating the rolling horizon, the second the true horizon and the third the node.

Be careful about the notation: $D_{T,k}$ denotes the *optimal* decision at node *k* for an optimising decision who has to liquidate in period *T*. In contrast $d_{S,T,k}$ denotes the decision at node *k* of a DM with a rolling horizon of *S* periods ahead in a problem where he/she actually has to liquidate in period *T* but wrongly working on the presumption that they have to liquidate *S* periods ahead.

It follows that we have the following results:

(18) If $t \geq T\text{-}S$ then $d_{S,T,k} = D_{T,k}$ because the true horizon is within the correct horizon.

(19) If $t < T\text{-}S$ then $d_{S,T,k} = D_{t+s,k}$ because the DM is optimising under the (wrong) assumption that he/she *has* to liquidate in period *t+S*.