L7 (By Peter Lohmander 2009-10-11)

Stochastic dynamic programming. General theory, application examples, analytical solutions and numerical solutions via computer programming.

Useful links:

Lohmander, P., Presentation of (stochastic and deterministic) dynamic programming:
http://www.lohmander.com/UPV/Lohmander_UPV_08.ppt
(Pages 3 – 11, 22 - 27)
The presentation covers two sections of the following text:

Lohmander, P., Adaptive Optimization of Forest Management in a Stochastic World, in Weintraub A. et al (Editors), Handbook of Operations Research in Natural Resources, Springer, Springer Science, International Series in Operations Research and Management Science, New York, USA, pp 525-544, 2007
http://www.amazon.ca/gp/reader/0387718141/ref=sib_dp_pt/701-0734992-1741115#reader-link

Lohmander, P., Optimal stock and purchase policy with stochastic external deliveries in different markets, Presented at the Association of American Foresters Systems Analysis Symposium, 2006
(Power Point Presentation, http://www.lohmander.com/Vermont2.ppt)

We start with a deterministic dynamic programming model (compare L6). The code is more compact than in L6 and the output has been designed to give tables that may be directly used in production planning.

```
! DP_CALC_091009;
! Peter Lohmander;

model:

sets:
time/1..11/:;
stindex/1..5/:stock;
timestock(time,stindex):f,optprod;
prindex/1..6/:prod,cost;
endsets

data:
stock = 0  1  2  3  4;
prod =  0  1  2  3  4  5;
cost =  0 15 17 19 21 23;
enddata

CALC:

TMAX = 11;
imax = 4;
xmax = 5;
h = 1;
d = 3;

@for(stindex(i): f(TMAX,i) = 0);
@for(stindex(i):optprod(TMAX,i) = 0);
```

```
t = tmax;
@while(t #GT#1:
t=t-1;
@write(@newline(1), 't = ', t);
@write(@newline(1), 'Entering Stock   Optimal Production   Total Cost   Entering Stock Next Period');

  i = imax+1;
  @while(i #GT#0:
     i = i-1;
     fopt = 9999999;
     xopt = 99;

     x = xmax+1;
     @while(x #GT#0:
            x = x-1;
            prodindex = x+1;
            stocknext = i+x-d;
            indexnext = stocknext+1;
            fev = 9999999;
         @IFC( indexnext#GE#1 #AND# indexnext#LE#5 :
               fev = cost(prodindex) + h*(i+x-d) + f(t+1,indexnext)
         );
         @IFC( indexnext#GE#1 #AND# indexnext#LE#5 #AND# fev#LT#fopt :
               xopt = x;
               fopt = fev;
               stocknopt = stocknext;
         );
         f(t,i+1) = fopt;
         optprod(t,i+1) = xopt;
  );
@write(@newline(1), (@FORMAT( i, '14.0f')), (@FORMAT(xopt , '21.0f')),
      (@FORMAT( fopt, '13.0f')), (@FORMAT( stocknopt, '29.0f'))) ;
);
@write(@newline(1));
);
@write(@newline(1));
ENDCALC
end
```

t = 10

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	1	1
3	0	0	0
2	1	15	0
1	2	17	0
0	3	19	0

t = 9

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	18	1
3	0	19	0
2	4	24	3
1	5	26	3
0	3	38	0

t = 8

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	27	1
3	0	38	0
2	4	43	3
1	5	45	3
0	4	48	1

t = 7

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	46	1
3	0	48	0
2	5	54	4
1	5	64	3
0	4	67	1

t = 6

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	65	1
3	0	67	0
2	4	72	3
1	5	74	3
0	5	79	2

t = 5

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	75	1
3	0	79	0
2	4	91	3
1	5	93	3
0	4	96	1

t = 4

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	94	1
3	0	96	0
2	5	102	4
1	5	105	3
0	4	115	1

t = 3

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	106	1
3	0	115	0
2	4	120	3
1	5	122	3
0	5	127	2

t = 2

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	123	1
3	0	127	0
2	5	133	4
1	5	141	3
0	4	144	1

t = 1

Entering Stock	Optimal Production	Total Cost	Entering Stock Next Period
4	0	142	1
3	0	144	0
2	5	150	4
1	5	153	3
0	5	158	2

Stochastic dynamic programming in finite time

Here, a stochastic dynamic programming model is presented.
A similar model with empirical background described in Swedish can be studied here:

http://www.lohmander.com/LagerB/LagerB.pdf
and here
http://www.lohmander.com/LagerB/LagerB.doc

CASE 0: STANDARD

```
stst_a

REM stst1.bas
REM Peter Lohmander 20081101

OPEN "ststut.dat" FOR OUTPUT AS #1

DIM f(30, 30), s(30), qopt(30, 30)

k = 4
a = 600
b = 300
r = .07
h = 10
tmax = 20
imax = 15
p0 = 400
p1 = 0
d = EXP(-r / 52)

ttabmax = 30
itabmax = 30


FOR i = 0 TO k
s(i) = k - i
NEXT i

FOR i = k + 1 TO itabmax
s(i) = 0
NEXT i
```

```
FOR t = 0 TO ttabmax
FOR i = 0 TO itabmax
f(t, i) = 0
qopt(t, i) = 0
NEXT i
NEXT t

FOR i = 0 TO k
f(tmax, i) = a * s(i)
qopt(tmax, i) = 0
NEXT i

FOR i = k + 1 TO itabmax
f(tmax, i) = -b * (i - k)
NEXT i


FOR t = tmax - 1 TO 0 STEP -1

FOR i = 0 TO imax
fopt = 99999
qopt(t, i) = 0


qmax = imax - i - 4
IF (qmax < 0) THEN qmax = 0

FOR q = 0 TO qmax

fev = a * s(i) + (p0 + p1 * q) * q

fev = fev + h * (i + s(i) - k + q)

fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 0)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 1)))
fev = fev + d * (.4 * f(t + 1, (i + s(i) - k + q + 2)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 3)))
fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 4)))


IF (fev < fopt) THEN qopt(t, i) = q
IF (fev < fopt) THEN f(t, i) = fev
IF (fev < fopt) THEN fopt = fev

NEXT q
NEXT i
NEXT t

REM ------------------------------------------------------------------
CLS
```

```
PRINT #1, "Optimal stock investment table"
PRINT #1, "****************************"

PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; qopt(t, i);

NEXT i
PRINT #1, ""
NEXT t

REM ------------------------------------------------------------------
PRINT #1, ""
PRINT #1, ""
PRINT #1, "Optimal present value table"
PRINT #1, "****************************"
PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; f(t, i) / 1;

NEXT i
PRINT #1, ""
NEXT t

CLOSE ststut.dat

END
```

Optimal stock investment table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 1	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 2	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 3	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 4	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 5	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 6	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 7	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 8	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 9	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 10	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 11	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 12	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 13	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 14	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 15	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 16	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 17	4	4	4	4	4	3	2	1	0	0	0	0	0	0	0	0
t = 18	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 19	2	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0
t = 20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Optimal present value table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	19116	18516	17916	17316	16716	16316	15916	15516	15116	14728	14342	13962	13587	13218	12854	12495
t = 1	18296	17696	17096	16496	15896	15496	15096	14696	14296	13908	13522	13142	12768	12398	12034	11675
t = 2	17476	16876	16276	15676	15076	14676	14276	13876	13476	13087	12702	12322	11947	11577	11213	10854
t = 3	16654	16054	15454	14854	14254	13854	13454	13054	12654	12265	11880	11500	11125	10755	10391	10032
t = 4	15831	15231	14631	14031	13431	13031	12631	12231	11831	11442	11057	10677	10302	9932	9568	9209
t = 5	15006	14406	13806	13206	12606	12206	11806	11406	11006	10618	10233	9853	9478	9108	8744	8385
t = 6	14181	13581	12981	12381	11781	11381	10981	10581	10181	9793	9407	9027	8652	8283	7919	7560
t = 7	13355	12755	12155	11555	10955	10555	10155	9755	9355	8967	8581	8201	7826	7457	7092	6734
t = 8	12527	11927	11327	10727	10127	9727	9327	8927	8527	8139	7753	7374	6999	6629	6265	5906
t = 9	11699	11099	10499	9899	9299	8899	8499	8099	7699	7311	6925	6545	6170	5801	5436	5078
t = 10	10869	10269	9669	9069	8469	8069	7669	7269	6869	6481	6095	5715	5340	4971	4607	4248
t = 11	10038	9438	8838	8238	7638	7238	6838	6438	6038	5650	5264	4885	4510	4140	3776	3417
t = 12	9207	8607	8007	7407	6807	6407	6007	5607	5207	4818	4433	4053	3678	3308	2945	2587
t = 13	8373	7773	7173	6573	5973	5573	5173	4773	4373	3985	3599	3220	2845	2476	2113	1758
t = 14	7539	6939	6339	5739	5139	4739	4339	3939	3539	3151	2765	2386	2012	1645	1287	942
t = 15	6704	6104	5504	4904	4304	3904	3504	3104	2704	2316	1930	1552	1182	824	482	161
t = 16	5868	5268	4668	4068	3468	3068	2668	2268	1868	1480	1096	724	369	37	−270	−556
t = 17	5030	4430	3830	3230	2630	2230	1830	1430	1030	646	274	−70	−385	−676	−953	−1224
t = 18	4191	3591	2991	2391	1791	1391	991	591	194	−160	−480	−775	−1057	−1336	−1615	−1894
t = 19	3340	2740	2140	1540	940	540	140	−240	−559	−849	−1138	−1428	−1718	−2007	−2297	−2586
t = 20	2400	1800	1200	600	0	−300	−600	−900	−1200	−1500	−1800	−2100	−2400	−2700	−3000	−3300

CASE 1: Increased storage costs per stored unit

```
stst_b

REM stst_b.bas
REM Peter Lohmander 20081101

OPEN "ststut.dat" FOR OUTPUT AS #1

DIM f(30, 30), s(30), qopt(30, 30)

k = 4
a = 600
b = 300
r = .07
h = 20
tmax = 20
imax = 15
p0 = 400
p1 = 0
d = EXP(-r / 52)

ttabmax = 30
itabmax = 30

FOR i = 0 TO k
s(i) = k - i
NEXT i

FOR i = k + 1 TO itabmax
s(i) = 0
NEXT i

FOR t = 0 TO ttabmax
FOR i = 0 TO itabmax
f(t, i) = 0
qopt(t, i) = 0
NEXT i
NEXT t

FOR i = 0 TO k
f(tmax, i) = a * s(i)
qopt(tmax, i) = 0
NEXT i

FOR i = k + 1 TO itabmax
f(tmax, i) = -b * (i - k)
NEXT i
```

```
FOR t = tmax - 1 TO 0 STEP -1

FOR i = 0 TO imax
fopt = 99999
qopt(t, i) = 0


qmax = imax - i - 4
IF (qmax < 0) THEN qmax = 0

FOR q = 0 TO qmax

fev = a * s(i) + (p0 + p1 * q) * q

fev = fev + h * (i + s(i) - k + q)

fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 0)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 1)))
fev = fev + d * (.4 * f(t + 1, (i + s(i) - k + q + 2)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 3)))
fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 4)))


IF (fev < fopt) THEN qopt(t, i) = q
IF (fev < fopt) THEN f(t, i) = fev
IF (fev < fopt) THEN fopt = fev

NEXT q
NEXT i
NEXT t

REM ------------------------------------------------------------------
CLS

PRINT #1, "Optimal stock investment table"
PRINT #1, "*****************************"

PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; qopt(t, i);

NEXT i
PRINT #1, ""
```

```
NEXT t

REM ------------------------------------------------------------------
PRINT #1, ""
PRINT #1, ""
PRINT #1, "Optimal present value table"
PRINT #1, "****************************"
PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; f(t, i) / 1;

NEXT i
PRINT #1, ""
NEXT t

CLOSE ststut.dat

END
```

Optimal stock investment table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 1	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 2	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 3	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 4	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 5	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 6	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 7	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 8	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 9	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 10	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 11	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 12	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 13	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 14	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 15	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 16	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 17	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 18	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 19	2	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0
t = 20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Optimal present value table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	19867	19267	18667	18067	17467	17067	16667	16267	15868	15491	15119	14758	14408	14068	13738	13418
t = 1	19009	18409	17809	17209	16609	16209	15809	15409	15010	14633	14261	13900	13550	13210	12879	12560
t = 2	18150	17550	16950	16350	15750	15350	14950	14550	14150	13773	13402	13041	12690	12350	12020	11700
t = 3	17289	16689	16089	15489	14889	14489	14089	13689	13290	12913	12541	12180	11830	11490	11160	10840
t = 4	16428	15828	15228	14628	14028	13628	13228	12828	12428	12051	11679	11319	10968	10628	10298	9978
t = 5	15565	14965	14365	13765	13165	12765	12365	11965	11565	11188	10817	10456	10105	9765	9435	9115
t = 6	14701	14101	13501	12901	12301	11901	11501	11101	10702	10324	9953	9592	9241	8901	8571	8251
t = 7	13836	13236	12636	12036	11436	11036	10636	10236	9836	9459	9088	8727	8376	8036	7706	7386
t = 8	12969	12369	11769	11169	10569	10169	9769	9369	8970	8593	8221	7861	7510	7170	6840	6520
t = 9	12102	11502	10902	10302	9702	9302	8902	8502	8103	7726	7354	6993	6642	6302	5972	5653
t = 10	11233	10633	10033	9433	8833	8433	8033	7633	7234	6857	6485	6125	5774	5434	5104	4784
t = 11	10364	9764	9164	8564	7964	7564	7164	6764	6364	5987	5615	5255	4904	4564	4234	3915
t = 12	9493	8893	8293	7693	7093	6693	6293	5893	5493	5116	4744	4384	4033	3693	3364	3045
t = 13	8620	8020	7420	6820	6220	5820	5420	5020	4621	4244	3872	3512	3161	2822	2494	2180
t = 14	7747	7147	6547	5947	5347	4947	4547	4147	3748	3371	2999	2639	2290	1953	1632	1329
t = 15	6873	6273	5673	5073	4473	4073	3673	3273	2873	2496	2125	1767	1423	1098	794	517
t = 16	5997	5397	4797	4197	3597	3197	2797	2397	1998	1622	1254	905	579	281	11	−237
t = 17	5120	4520	3920	3320	2720	2320	1920	1520	1122	752	401	83	−204	−466	−713	−954
t = 18	4242	3642	3042	2442	1842	1442	1042	642	258	−79	−381	−655	−917	−1176	−1435	−1695
t = 19	3360	2760	2160	1560	960	560	160	−210	−519	−799	−1078	−1358	−1638	−1917	−2197	−2476
t = 20	2400	1800	1200	600	0	−300	−600	−900	−1200	−1500	−1800	−2100	−2400	−2700	−3000	−3300

CASE 2: Increased slope of the supply function

```
stst_c

REM stst_c.bas
REM Peter Lohmander 20081101

OPEN "ststut.dat" FOR OUTPUT AS #1

DIM f(30, 30), s(30), qopt(30, 30)

k = 4
a = 600
b = 300
r = .07
h = 10
tmax = 20
imax = 15
p0 = 400
p1 = 10
d = EXP(-r / 52)

ttabmax = 30
itabmax = 30

FOR i = 0 TO k
s(i) = k - i
NEXT i

FOR i = k + 1 TO itabmax
s(i) = 0
NEXT i

FOR t = 0 TO ttabmax
FOR i = 0 TO itabmax
f(t, i) = 0
qopt(t, i) = 0
NEXT i
NEXT t

FOR i = 0 TO k
f(tmax, i) = a * s(i)
qopt(tmax, i) = 0
NEXT i

FOR i = k + 1 TO itabmax
f(tmax, i) = -b * (i - k)
NEXT i
```

```
FOR t = tmax - 1 TO 0 STEP -1

FOR i = 0 TO imax
fopt = 99999
qopt(t, i) = 0

qmax = imax - i - 4
IF (qmax < 0) THEN qmax = 0

FOR q = 0 TO qmax

fev = a * s(i) + (p0 + p1 * q) * q

fev = fev + h * (i + s(i) - k + q)

fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 0)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 1)))
fev = fev + d * (.4 * f(t + 1, (i + s(i) - k + q + 2)))
fev = fev + d * (.2 * f(t + 1, (i + s(i) - k + q + 3)))
fev = fev + d * (.1 * f(t + 1, (i + s(i) - k + q + 4)))

IF (fev < fopt) THEN qopt(t, i) = q
IF (fev < fopt) THEN f(t, i) = fev
IF (fev < fopt) THEN fopt = fev

NEXT q
NEXT i
NEXT t

REM ------------------------------------------------------------------
CLS
PRINT #1, "Optimal stock investment table"
PRINT #1, "******************************"

PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; qopt(t, i);

NEXT i
PRINT #1, ""
NEXT t

REM ------------------------------------------------------------------
```

```
PRINT #1, ""
PRINT #1, ""
PRINT #1, "Optimal present value table"
PRINT #1, "*****************************"
PRINT #1, "i =           0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15"
PRINT #1, ""
FOR t = 0 TO tmax

PRINT #1, "t = ";
PRINT #1, USING "##"; t;
PRINT #1, "  ";

FOR i = 0 TO imax
PRINT #1, USING "#######"; f(t, i) / 1;

NEXT i
PRINT #1, ""
NEXT t

CLOSE ststut.dat

END
```

Optimal stock investment table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 1	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 2	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 3	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 4	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 5	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 6	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 7	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 8	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 9	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 10	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 11	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 12	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 13	3	3	3	3	3	2	2	1	1	1	0	0	0	0	0	0
t = 14	3	3	3	3	3	2	2	1	1	0	0	0	0	0	0	0
t = 15	3	3	3	3	3	2	2	1	1	0	0	0	0	0	0	0
t = 16	3	3	3	3	3	2	2	1	1	0	0	0	0	0	0	0
t = 17	3	3	3	3	3	2	2	1	0	0	0	0	0	0	0	0
t = 18	3	3	3	3	3	2	1	0	0	0	0	0	0	0	0	0
t = 19	2	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0
t = 20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Optimal present value table

i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t = 0	20083	19483	18883	18283	17683	17233	16787	16357	15936	15524	15114	14710	14312	13919	13531	13148
t = 1	19216	18616	18016	17416	16816	16366	15921	15491	15070	14658	14248	13844	13446	13053	12665	12282
t = 2	18349	17749	17149	16549	15949	15499	15053	14623	14202	13790	13380	12977	12578	12185	11797	11414
t = 3	17480	16880	16280	15680	15080	14630	14185	13755	13334	12922	12512	12108	11710	11317	10929	10546
t = 4	16611	16011	15411	14811	14211	13761	13315	12885	12464	12052	11642	11238	10840	10447	10059	9676
t = 5	15740	15140	14540	13940	13340	12890	12444	12014	11593	11181	10771	10368	9969	9576	9188	8805
t = 6	14868	14268	13668	13068	12468	12018	11572	11142	10721	10309	9899	9496	9097	8704	8316	7933
t = 7	13994	13394	12794	12194	11594	11144	10699	10269	9848	9436	9026	8622	8224	7831	7443	7060
t = 8	13120	12520	11920	11320	10720	10270	9824	9394	8973	8562	8152	7748	7350	6957	6569	6186
t = 9	12244	11644	11044	10444	9844	9394	8949	8519	8098	7686	7276	6873	6474	6081	5693	5311
t = 10	11368	10768	10168	9568	8968	8518	8072	7642	7221	6809	6399	5996	5598	5205	4817	4435
t = 11	10490	9890	9290	8690	8090	7640	7194	6764	6343	5932	5522	5118	4720	4328	3940	3559
t = 12	9610	9010	8410	7810	7210	6760	6315	5885	5464	5053	4643	4240	3842	3450	3064	2684
t = 13	8730	8130	7530	6930	6330	5880	5435	5005	4584	4174	3764	3361	2964	2573	2190	1816
t = 14	7848	7248	6648	6048	5448	4998	4553	4123	3704	3294	2884	2482	2087	1701	1327	968
t = 15	6964	6364	5764	5164	4564	4114	3671	3241	2823	2413	2005	1607	1219	847	495	168
t = 16	6079	5479	4879	4279	3679	3229	2789	2359	1943	1533	1132	745	380	42	−269	−556
t = 17	5190	4590	3990	3390	2790	2340	1906	1476	1066	664	283	−66	−384	−676	−953	−1224
t = 18	4297	3697	3097	2497	1897	1447	1017	607	200	−159	−480	−775	−1057	−1336	−1615	−1894
t = 19	3380	2780	2180	1580	980	550	140	−240	−559	−849	−1138	−1428	−1718	−2007	−2297	−2586
t = 20	2400	1800	1200	600	0	−300	−600	−900	−1200	−1500	−1800	−2100	−2400	−2700	−3000	−3300

Deterministic dynamic programming in infinite time

We start with a fundamental deterministic dynamic programming problem in infinite time:

```
! Dynopt1_080916  ;
! Peter Lohmander ;

min = w1 + w2;

d = .9;

[u11] w1 >= 5 + d*w1;

[u12] w1 >= 0 + d*w2;

[u21] w2 >= 14 + d*w1;

[u22] w2 >= 6 + d*w2;
end
```

```
  Global optimal solution found.
  Objective value:                              140.0000
  Infeasibilities:                              0.000000
  Total solver iterations:                             0

                       Variable           Value        Reduced Cost
                             W1        66.31579            0.000000
                             W2        73.68421            0.000000
                              D       0.9000000            0.000000

                            Row    Slack or Surplus      Dual Price
                              1        140.0000           -1.000000
                              2        0.000000           -1400.000
                            U11        1.631579            0.000000
                            U12        0.000000           -10.00000
                            U21        0.000000           -10.00000
                            U22        1.368421            0.000000
```

Stochastic dynamic programming in infinite time

Power point presentation:
Lohmander, P., Optimal stock and purchase policy with stochastic external deliveries in different markets, Presented at Association of American Foresters Systems Analysis Symposium, 2006
(Power Point Presentation, http://www.lohmander.com/Vermont2.ppt)

Stochastic dynamic programming in infinite time – with a perfect raw material market

060614 1511 P = 330 + 0*u
(No stock level increase)

```
! Vermont06_060614;
! Peter Lohmander;

sets:
s_set/1..21/:s,w;
u_set/1..7/:u,c;
b_set/1..7/:dev,pdev;
q_set/1..7/:q, Rev;
su_set(s_set,u_set): prod;
endsets

sumprob = @sum(b_set(i):pdev);

!Parameters;
!**********;
smax = 21;
millcap = 6;
csetup = 0;
margprof = 1000;
```

```
mcstock = 2;
p0 = 330;
p1 = 0;
r = .10;
dyear = 1/(1+r);
d = dyear^(1/365);


@FOR( s_set(i): s(i) = i-1 );

@FOR( u_set(i): u(i) = i-1 );

@for(u_set(i): c(i) = (p0+p1*u(i))*u(i));

@FOR( b_set(i): dev(i) = i-4 );
@for( b_set(i):@free(dev(i)));

pdev(1) = 1/64;
pdev(2) = 6/64;
pdev(3) = 15/64;
pdev(4) = 20/64;
pdev(5) = 15/64;
pdev(6) = 6/64;
pdev(7) = 1/64;

@FOR( q_set(i): q(i) = i-1);
@FOR( q_set(i)|q(i)#LT#1 : Rev(i) = 0);
@FOR( q_set(i)|q(i)#GE#1 : Rev(i) = -csetup + margprof*q(i));


obj =  @SUM( s_set(i)|i#LE# (smax-10):w(i));
min = obj;
```

```
@FOR( s_set(i):
  @FOR( u_set(j):
    prod(i,j) = @SMIN( millcap, (s(i)+ u(j)))
));


@FOR( s_set(i):
  @FOR( u_set(j)| prod(i,j)#LE# millcap #AND# (i+u(j)-prod(i,j))#LE#(smax-10):
  [w_]   w(i) >= Rev(1+prod(i,j)) - c(j) - mcstock*(s(i)+u(j)-prod(i,j))
           + d*(pdev(1)*w(i+u(j)-prod(i,j) + 0) +
                pdev(2)*w(i+u(j)-prod(i,j) + 1) +
                pdev(3)*w(i+u(j)-prod(i,j) + 2) +
                pdev(4)*w(i+u(j)-prod(i,j) + 3) +
                pdev(5)*w(i+u(j)-prod(i,j) + 4) +
                pdev(6)*w(i+u(j)-prod(i,j) + 5) +
                pdev(7)*w(i+u(j)-prod(i,j) + 6) )
));


@FOR( s_set(i) | i#GT#(smax-10):
    w(i) = w(smax-10) +
          (i-(smax-10))*(w(smax-10) - w(smax-15))/5 );

@for(s_set(i):@free(w(i)));

end

------------------------------------------------------------------------------

 Rows=    110 Vars=     22 No. integer vars=      0  ( all are linear)
```

Nonzeros= 826 Constraint nonz= 728(23 are +- 1) Density=0.326
Smallest and largest elements in abs value= 0.156209E-01 6000.00
No. < : 0 No. =: 11 No. > : 98, Obj=MIN, GUBs <= 7
Single cols= 4

Global optimal solution found at step: 230
Objective value: 0.2110841E+09

Variable	Value	Reduced Cost
SUMPROB	1.000000	0.0000000
SMAX	21.00000	0.0000000
MILLCAP	6.000000	0.0000000
CSETUP	0.0000000	0.0000000
MARGPROF	1000.000	0.0000000
MCSTOCK	2.000000	0.0000000
P0	330.0000	0.0000000
P1	0.0000000	0.0000000
R	0.1000000	0.0000000
DYEAR	0.9090909	0.0000000
D	0.9997389	0.0000000
OBJ	0.2110841E+09	0.0000000
S(1)	0.0000000	0.0000000
S(2)	1.000000	0.0000000
S(3)	2.000000	0.0000000
S(4)	3.000000	0.0000000
S(5)	4.000000	0.0000000
S(6)	5.000000	0.0000000
S(7)	6.000000	0.0000000
S(8)	7.000000	0.0000000
S(9)	8.000000	0.0000000
S(10)	9.000000	0.0000000
S(11)	10.00000	0.0000000

S(12)	11.00000	0.0000000
S(13)	12.00000	0.0000000
S(14)	13.00000	0.0000000
S(15)	14.00000	0.0000000
S(16)	15.00000	0.0000000
S(17)	16.00000	0.0000000
S(18)	17.00000	0.0000000
S(19)	18.00000	0.0000000
S(20)	19.00000	0.0000000
S(21)	20.00000	0.0000000
W(1)	0.1918782E+08	0.0000000
W(2)	0.1918815E+08	0.0000000
W(3)	0.1918848E+08	0.0000000
W(4)	0.1918881E+08	0.0000000
W(5)	0.1918914E+08	0.0000000
W(6)	0.1918947E+08	0.0000000
W(7)	0.1918980E+08	0.0000000
W(8)	0.1919013E+08	0.0000000
W(9)	0.1919045E+08	0.0000000
W(10)	0.1919078E+08	0.0000000
W(11)	0.1919111E+08	0.0000000
W(12)	0.1919144E+08	0.0000000
W(13)	0.1919176E+08	0.0000000
W(14)	0.1919209E+08	0.0000000
W(15)	0.1919242E+08	0.0000000
W(16)	0.1919275E+08	0.0000000
W(17)	0.1919307E+08	0.0000000
W(18)	0.1919340E+08	0.0000000
W(19)	0.1919373E+08	0.0000000
W(20)	0.1919406E+08	0.0000000
W(21)	0.1919439E+08	0.0000000
U(1)	0.0000000	0.0000000

U(2)	1.000000	0.0000000
U(3)	2.000000	0.0000000
U(4)	3.000000	0.0000000
U(5)	4.000000	0.0000000
U(6)	5.000000	0.0000000
U(7)	6.000000	0.0000000
C(1)	0.0000000	0.0000000
C(2)	330.0000	0.0000000
C(3)	660.0000	0.0000000
C(4)	990.0000	0.0000000
C(5)	1320.000	0.0000000
C(6)	1650.000	0.0000000
C(7)	1980.000	0.0000000
DEV(1)	-3.000000	0.0000000
DEV(2)	-2.000000	0.0000000
DEV(3)	-1.000000	0.0000000
DEV(4)	0.0000000	0.0000000
DEV(5)	1.000000	0.0000000
DEV(6)	2.000000	0.0000000
DEV(7)	3.000000	0.0000000
PDEV(1)	0.1562500E-01	0.0000000
PDEV(2)	0.9375000E-01	0.0000000
PDEV(3)	0.2343750	0.0000000
PDEV(4)	0.3125000	0.0000000
PDEV(5)	0.2343750	0.0000000
PDEV(6)	0.9375000E-01	0.0000000
PDEV(7)	0.1562500E-01	0.0000000
Q(1)	0.0000000	0.0000000
Q(2)	1.000000	0.0000000
Q(3)	2.000000	0.0000000
Q(4)	3.000000	0.0000000
Q(5)	4.000000	0.0000000

Q(6)	5.000000	0.0000000
Q(7)	6.000000	0.0000000
REV(1)	0.0000000	0.0000000
REV(2)	1000.000	0.0000000
REV(3)	2000.000	0.0000000
REV(4)	3000.000	0.0000000
REV(5)	4000.000	0.0000000
REV(6)	5000.000	0.0000000
REV(7)	6000.000	0.0000000
PROD(1, 1)	0.2441406E-03	0.0000000
PROD(1, 2)	1.000000	0.0000000
PROD(1, 3)	2.000000	0.0000000
PROD(1, 4)	3.000000	0.0000000
PROD(1, 5)	4.000000	0.0000000
PROD(1, 6)	5.000000	0.0000000
PROD(1, 7)	6.000000	0.0000000
PROD(2, 1)	1.000000	0.0000000
PROD(2, 2)	2.000000	0.0000000
PROD(2, 3)	3.000000	0.0000000
PROD(2, 4)	4.000000	0.0000000
PROD(2, 5)	5.000000	0.0000000
PROD(2, 6)	6.000000	0.0000000
PROD(2, 7)	6.000000	0.0000000
PROD(3, 1)	2.000000	0.0000000
PROD(3, 2)	3.000000	0.0000000
PROD(3, 3)	4.000000	0.0000000
PROD(3, 4)	5.000000	0.0000000
PROD(3, 5)	6.000000	0.0000000
PROD(3, 6)	6.000000	0.0000000
PROD(3, 7)	6.000000	0.0000000
PROD(4, 1)	3.000000	0.0000000
PROD(4, 2)	4.000000	0.0000000

PROD(4, 3)	5.000000	0.0000000
PROD(4, 4)	6.000000	0.0000000
PROD(4, 5)	6.000000	0.0000000
PROD(4, 6)	6.000000	0.0000000
PROD(4, 7)	6.000000	0.0000000
PROD(5, 1)	4.000000	0.0000000
PROD(5, 2)	5.000000	0.0000000
PROD(5, 3)	6.000000	0.0000000
PROD(5, 4)	6.000000	0.0000000
PROD(5, 5)	6.000000	0.0000000
PROD(5, 6)	6.000000	0.0000000
PROD(5, 7)	6.000000	0.0000000
PROD(6, 1)	5.000000	0.0000000
PROD(6, 2)	6.000000	0.0000000
PROD(6, 3)	6.000000	0.0000000
PROD(6, 4)	6.000000	0.0000000
PROD(6, 5)	6.000000	0.0000000
PROD(6, 6)	6.000000	0.0000000
PROD(6, 7)	6.000000	0.0000000
PROD(7, 1)	6.000000	0.0000000
PROD(7, 2)	6.000000	0.0000000
PROD(7, 3)	6.000000	0.0000000
PROD(7, 4)	6.000000	0.0000000
PROD(7, 5)	6.000000	0.0000000
PROD(7, 6)	6.000000	0.0000000
PROD(7, 7)	6.000000	0.0000000
PROD(8, 1)	6.000000	0.0000000
PROD(8, 2)	6.000000	0.0000000
PROD(8, 3)	6.000000	0.0000000
PROD(8, 4)	6.000000	0.0000000
PROD(8, 5)	6.000000	0.0000000
PROD(8, 6)	6.000000	0.0000000

PROD(8, 7)	6.000000	0.0000000
PROD(9, 1)	6.000000	0.0000000
PROD(9, 2)	6.000000	0.0000000
PROD(9, 3)	6.000000	0.0000000
PROD(9, 4)	6.000000	0.0000000
PROD(9, 5)	6.000000	0.0000000
PROD(9, 6)	6.000000	0.0000000
PROD(9, 7)	6.000000	0.0000000
PROD(10, 1)	6.000000	0.0000000
PROD(10, 2)	6.000000	0.0000000
PROD(10, 3)	6.000000	0.0000000
PROD(10, 4)	6.000000	0.0000000
PROD(10, 5)	6.000000	0.0000000
PROD(10, 6)	6.000000	0.0000000
PROD(10, 7)	6.000000	0.0000000
PROD(11, 1)	6.000000	0.0000000
PROD(11, 2)	6.000000	0.0000000
PROD(11, 3)	6.000000	0.0000000
PROD(11, 4)	6.000000	0.0000000
PROD(11, 5)	6.000000	0.0000000
PROD(11, 6)	6.000000	0.0000000
PROD(11, 7)	6.000000	0.0000000
PROD(12, 1)	6.000000	0.0000000
PROD(12, 2)	6.000000	0.0000000
PROD(12, 3)	6.000000	0.0000000
PROD(12, 4)	6.000000	0.0000000
PROD(12, 5)	6.000000	0.0000000
PROD(12, 6)	6.000000	0.0000000
PROD(12, 7)	6.000000	0.0000000
PROD(13, 1)	6.000000	0.0000000
PROD(13, 2)	6.000000	0.0000000
PROD(13, 3)	6.000000	0.0000000

PROD(13, 4)	6.000000	0.0000000
PROD(13, 5)	6.000000	0.0000000
PROD(13, 6)	6.000000	0.0000000
PROD(13, 7)	6.000000	0.0000000
PROD(14, 1)	6.000000	0.0000000
PROD(14, 2)	6.000000	0.0000000
PROD(14, 3)	6.000000	0.0000000
PROD(14, 4)	6.000000	0.0000000
PROD(14, 5)	6.000000	0.0000000
PROD(14, 6)	6.000000	0.0000000
PROD(14, 7)	6.000000	0.0000000
PROD(15, 1)	6.000000	0.0000000
PROD(15, 2)	6.000000	0.0000000
PROD(15, 3)	6.000000	0.0000000
PROD(15, 4)	6.000000	0.0000000
PROD(15, 5)	6.000000	0.0000000
PROD(15, 6)	6.000000	0.0000000
PROD(15, 7)	6.000000	0.0000000
PROD(16, 1)	6.000000	0.0000000
PROD(16, 2)	6.000000	0.0000000
PROD(16, 3)	6.000000	0.0000000
PROD(16, 4)	6.000000	0.0000000
PROD(16, 5)	6.000000	0.0000000
PROD(16, 6)	6.000000	0.0000000
PROD(16, 7)	6.000000	0.0000000
PROD(17, 1)	6.000000	0.0000000
PROD(17, 2)	6.000000	0.0000000
PROD(17, 3)	6.000000	0.0000000
PROD(17, 4)	6.000000	0.0000000
PROD(17, 5)	6.000000	0.0000000
PROD(17, 6)	6.000000	0.0000000
PROD(17, 7)	6.000000	0.0000000

PROD(18, 1)	6.000000	0.0000000
PROD(18, 2)	6.000000	0.0000000
PROD(18, 3)	6.000000	0.0000000
PROD(18, 4)	6.000000	0.0000000
PROD(18, 5)	6.000000	0.0000000
PROD(18, 6)	6.000000	0.0000000
PROD(18, 7)	6.000000	0.0000000
PROD(19, 1)	6.000000	0.0000000
PROD(19, 2)	6.000000	0.0000000
PROD(19, 3)	6.000000	0.0000000
PROD(19, 4)	6.000000	0.0000000
PROD(19, 5)	6.000000	0.0000000
PROD(19, 6)	6.000000	0.0000000
PROD(19, 7)	6.000000	0.0000000
PROD(20, 1)	6.000000	0.0000000
PROD(20, 2)	6.000000	0.0000000
PROD(20, 3)	6.000000	0.0000000
PROD(20, 4)	6.000000	0.0000000
PROD(20, 5)	6.000000	0.0000000
PROD(20, 6)	6.000000	0.0000000
PROD(20, 7)	6.000000	0.0000000
PROD(21, 1)	6.000000	0.0000000
PROD(21, 2)	6.000000	0.0000000
PROD(21, 3)	6.000000	0.0000000
PROD(21, 4)	6.000000	0.0000000
PROD(21, 5)	6.000000	0.0000000
PROD(21, 6)	6.000000	0.0000000
PROD(21, 7)	6.000000	0.0000000

Row	Slack or Surplus	Dual Price
W_(1, 1)	4020.000	0.0000000
W_(1, 2)	3350.000	0.0000000

W_(1, 3)	2680.000	0.0000000
W_(1, 4)	2010.000	0.0000000
W_(1, 5)	1340.000	0.0000000
W_(1, 6)	670.0000	0.0000000
W_(1, 7)	0.0000000	-659.0452
W_(2, 1)	3350.000	0.0000000
W_(2, 2)	2680.000	0.0000000
W_(2, 3)	2010.000	0.0000000
W_(2, 4)	1340.000	0.0000000
W_(2, 5)	670.0000	0.0000000
W_(2, 6)	0.0000000	-3949.298
W_(2, 7)	2.119264	0.0000000
W_(3, 1)	2680.000	0.0000000
W_(3, 2)	2010.000	0.0000000
W_(3, 3)	1340.000	0.0000000
W_(3, 4)	670.0000	0.0000000
W_(3, 5)	0.0000000	-9871.861
W_(3, 6)	2.119264	0.0000000
W_(3, 7)	4.440310	0.0000000
W_(4, 1)	2010.000	0.0000000
W_(4, 2)	1340.000	0.0000000
W_(4, 3)	670.0000	0.0000000
W_(4, 4)	0.0000000	-13162.46
W_(4, 5)	2.119264	0.0000000
W_(4, 6)	4.440310	0.0000000
W_(4, 7)	7.285021	0.0000000
W_(5, 1)	1340.000	0.0000000
W_(5, 2)	670.0000	0.0000000
W_(5, 3)	0.0000000	-9872.659
W_(5, 4)	2.119264	0.0000000
W_(5, 5)	4.440310	0.0000000
W_(5, 6)	7.285021	0.0000000

W_(5, 7)	10.90023	0.0000000
W_(6, 1)	670.0000	0.0000000
W_(6, 2)	0.0000000	-3950.459
W_(6, 3)	2.119264	0.0000000
W_(6, 4)	4.440310	0.0000000
W_(6, 5)	7.285021	0.0000000
W_(6, 6)	10.90023	0.0000000
W_(6, 7)	15.24754	0.0000000
W_(7, 1)	0.0000000	-660.1127
W_(7, 2)	2.119264	0.0000000
W_(7, 3)	4.440310	0.0000000
W_(7, 4)	7.285021	0.0000000
W_(7, 5)	10.90023	0.0000000
W_(7, 6)	15.24754	0.0000000
W_(7, 7)	20.05040	0.0000000
W_(8, 1)	0.0000000	-1.732947
W_(8, 2)	2.321046	0.0000000
W_(8, 3)	5.165756	0.0000000
W_(8, 4)	8.780961	0.0000000
W_(8, 5)	13.12828	0.0000000
W_(8, 6)	17.93113	0.0000000
W_(8, 7)	22.81314	0.0000000
W_(9, 1)	0.0000000	-1.363611
W_(9, 2)	2.844711	0.0000000
W_(9, 3)	6.459915	0.0000000
W_(9, 4)	10.80723	0.0000000
W_(9, 5)	15.61009	0.0000000
W_(9, 6)	20.49210	0.0000000
W_(9, 7)	25.15851	0.0000000
W_(10, 1)	0.0000000	-1.112593
W_(10, 2)	3.615205	0.0000000
W_(10, 3)	7.962521	0.0000000

W_(10, 4)	12.76538	0.0000000
W_(10, 5)	17.64739	0.0000000
W_(10, 6)	22.31380	0.0000000
W_(10, 7)	26.72433	0.0000000
W_(11, 1)	0.0000000	-1.015869
W_(11, 2)	4.347316	0.0000000
W_(11, 3)	9.150171	0.0000000
W_(11, 4)	14.03218	0.0000000
W_(11, 5)	18.69859	0.0000000
W_(11, 6)	23.10912	0.0000000
W_(11, 7)	27.39718	0.0000000
W_(12, 1)	2.167271	0.0000000
W_(12, 2)	6.970125	0.0000000
W_(12, 3)	11.85214	0.0000000
W_(12, 4)	16.51855	0.0000000
W_(12, 5)	20.92908	0.0000000
W_(12, 6)	25.21713	0.0000000
W_(13, 1)	4.790080	0.0000000
W_(13, 2)	9.672092	0.0000000
W_(13, 3)	14.33850	0.0000000
W_(13, 4)	18.74903	0.0000000
W_(13, 5)	23.03709	0.0000000
W_(14, 1)	7.492047	0.0000000
W_(14, 2)	12.15846	0.0000000
W_(14, 3)	16.56899	0.0000000
W_(14, 4)	20.85704	0.0000000
W_(15, 1)	9.978414	0.0000000
W_(15, 2)	14.38894	0.0000000
W_(15, 3)	18.67700	0.0000000
W_(16, 1)	12.20890	0.0000000
W_(16, 2)	16.49695	0.0000000
W_(17, 1)	14.31691	0.0000000

Stochastic dynamic programming in infinite time – with imperfect raw material market

060614 1518 P = 300 + 10*u
(Stock level increase)

```
! Vermont06_060614;
! Peter Lohmander;

sets:
s_set/1..21/:s,w;
u_set/1..7/:u,c;
b_set/1..7/:dev,pdev;
q_set/1..7/:q, Rev;
su_set(s_set,u_set): prod;
endsets

sumprob = @sum(b_set(i):pdev);

!Parameters;
!**********;
smax = 21;
millcap = 6;
csetup = 0;
margprof = 1000;
mcstock = 2;
p0 = 300;
p1 = 10;
r = .10;
dyear = 1/(1+r);
d = dyear^(1/365);

@FOR( s_set(i): s(i) = i-1 );
```

```
@FOR( u_set(i): u(i) = i-1 );

@for(u_set(i): c(i) = (p0+p1*u(i))*u(i));

@FOR( b_set(i): dev(i) = i-4 );
@for( b_set(i):@free(dev(i)));

pdev(1) = 1/64;
pdev(2) = 6/64;
pdev(3) = 15/64;
pdev(4) = 20/64;
pdev(5) = 15/64;
pdev(6) = 6/64;
pdev(7) = 1/64;

@FOR( q_set(i): q(i) = i-1);
@FOR( q_set(i)|q(i)#LT#1 : Rev(i) = 0);
@FOR( q_set(i)|q(i)#GE#1 : Rev(i) = -csetup + margprof*q(i));


obj =  @SUM( s_set(i)|i#LE# (smax-10):w(i));
min = obj;


@FOR( s_set(i):
  @FOR( u_set(j):
    prod(i,j) = @SMIN( millcap, (s(i)+ u(j)))
));


@FOR( s_set(i):
```

```
  @FOR( u_set(j)| prod(i,j)#LE# millcap #AND# (i+u(j)-prod(i,j))#LE#(smax-10):
  [w_]   w(i) >= Rev(1+prod(i,j)) - c(j) - mcstock*(s(i)+u(j)-prod(i,j))
           + d*(pdev(1)*w(i+u(j)-prod(i,j) + 0) +
                pdev(2)*w(i+u(j)-prod(i,j) + 1) +
                pdev(3)*w(i+u(j)-prod(i,j) + 2) +
                pdev(4)*w(i+u(j)-prod(i,j) + 3) +
                pdev(5)*w(i+u(j)-prod(i,j) + 4) +
                pdev(6)*w(i+u(j)-prod(i,j) + 5) +
                pdev(7)*w(i+u(j)-prod(i,j) + 6) )
));


@FOR( s_set(i) | i#GT#(smax-10):
    w(i) = w(smax-10) +
          (i-(smax-10))*(w(smax-10) - w(smax-15))/5 );

@for(s_set(i):@free(w(i)));

end

------------------------------------------------------------------------------

 Rows=    110 Vars=     22 No. integer vars=      0  ( all are linear)
 Nonzeros=    826 Constraint nonz=   728(    23 are +- 1) Density=0.326
 Smallest and largest elements in abs value=   0.156209E-01    6000.00
 No. < :   0 No. =:  11 No. > :  98, Obj=MIN, GUBs <=   7
 Single cols=    4

 Global optimal solution found at step:          253
 Objective value:                        0.2107713E+09

                     Variable           Value        Reduced Cost
```

SUMPROB	1.000000	0.0000000
SMAX	21.00000	0.0000000
MILLCAP	6.000000	0.0000000
CSETUP	0.0000000	0.0000000
MARGPROF	1000.000	0.0000000
MCSTOCK	2.000000	0.0000000
P0	300.0000	0.0000000
P1	10.00000	0.0000000
R	0.1000000	0.0000000
DYEAR	0.9090909	0.0000000
D	0.9997389	0.0000000
OBJ	0.2107713E+09	0.0000000
S(1)	0.0000000	0.0000000
S(2)	1.000000	0.0000000
S(3)	2.000000	0.0000000
S(4)	3.000000	0.0000000
S(5)	4.000000	0.0000000
S(6)	5.000000	0.0000000
S(7)	6.000000	0.0000000
S(8)	7.000000	0.0000000
S(9)	8.000000	0.0000000
S(10)	9.000000	0.0000000
S(11)	10.00000	0.0000000
S(12)	11.00000	0.0000000
S(13)	12.00000	0.0000000
S(14)	13.00000	0.0000000
S(15)	14.00000	0.0000000
S(16)	15.00000	0.0000000
S(17)	16.00000	0.0000000
S(18)	17.00000	0.0000000
S(19)	18.00000	0.0000000
S(20)	19.00000	0.0000000

S(21)	20.00000	0.0000000
W(1)	0.1915916E+08	0.0000000
W(2)	0.1915957E+08	0.0000000
W(3)	0.1915996E+08	0.0000000
W(4)	0.1916033E+08	0.0000000
W(5)	0.1916070E+08	0.0000000
W(6)	0.1916105E+08	0.0000000
W(7)	0.1916140E+08	0.0000000
W(8)	0.1916175E+08	0.0000000
W(9)	0.1916210E+08	0.0000000
W(10)	0.1916244E+08	0.0000000
W(11)	0.1916279E+08	0.0000000
W(12)	0.1916314E+08	0.0000000
W(13)	0.1916348E+08	0.0000000
W(14)	0.1916383E+08	0.0000000
W(15)	0.1916418E+08	0.0000000
W(16)	0.1916453E+08	0.0000000
W(17)	0.1916487E+08	0.0000000
W(18)	0.1916522E+08	0.0000000
W(19)	0.1916557E+08	0.0000000
W(20)	0.1916592E+08	0.0000000
W(21)	0.1916626E+08	0.0000000
U(1)	0.0000000	0.0000000
U(2)	1.000000	0.0000000
U(3)	2.000000	0.0000000
U(4)	3.000000	0.0000000
U(5)	4.000000	0.0000000
U(6)	5.000000	0.0000000
U(7)	6.000000	0.0000000
C(1)	0.0000000	0.0000000
C(2)	310.0000	0.0000000
C(3)	640.0000	0.0000000

C(4)	990.0000	0.0000000
C(5)	1360.000	0.0000000
C(6)	1750.000	0.0000000
C(7)	2160.000	0.0000000
DEV(1)	-3.000000	0.0000000
DEV(2)	-2.000000	0.0000000
DEV(3)	-1.000000	0.0000000
DEV(4)	0.0000000	0.0000000
DEV(5)	1.000000	0.0000000
DEV(6)	2.000000	0.0000000
DEV(7)	3.000000	0.0000000
PDEV(1)	0.1562500E-01	0.0000000
PDEV(2)	0.9375000E-01	0.0000000
PDEV(3)	0.2343750	0.0000000
PDEV(4)	0.3125000	0.0000000
PDEV(5)	0.2343750	0.0000000
PDEV(6)	0.9375000E-01	0.0000000
PDEV(7)	0.1562500E-01	0.0000000
Q(1)	0.0000000	0.0000000
Q(2)	1.000000	0.0000000
Q(3)	2.000000	0.0000000
Q(4)	3.000000	0.0000000
Q(5)	4.000000	0.0000000
Q(6)	5.000000	0.0000000
Q(7)	6.000000	0.0000000
REV(1)	0.0000000	0.0000000
REV(2)	1000.000	0.0000000
REV(3)	2000.000	0.0000000
REV(4)	3000.000	0.0000000
REV(5)	4000.000	0.0000000
REV(6)	5000.000	0.0000000
REV(7)	6000.000	0.0000000

PROD(1, 1)	0.2441406E-03	0.0000000
PROD(1, 2)	1.000000	0.0000000
PROD(1, 3)	2.000000	0.0000000
PROD(1, 4)	3.000000	0.0000000
PROD(1, 5)	4.000000	0.0000000
PROD(1, 6)	5.000000	0.0000000
PROD(1, 7)	6.000000	0.0000000
PROD(2, 1)	1.000000	0.0000000
PROD(2, 2)	2.000000	0.0000000
PROD(2, 3)	3.000000	0.0000000
PROD(2, 4)	4.000000	0.0000000
PROD(2, 5)	5.000000	0.0000000
PROD(2, 6)	6.000000	0.0000000
PROD(2, 7)	6.000000	0.0000000
PROD(3, 1)	2.000000	0.0000000
PROD(3, 2)	3.000000	0.0000000
PROD(3, 3)	4.000000	0.0000000
PROD(3, 4)	5.000000	0.0000000
PROD(3, 5)	6.000000	0.0000000
PROD(3, 6)	6.000000	0.0000000
PROD(3, 7)	6.000000	0.0000000
PROD(4, 1)	3.000000	0.0000000
PROD(4, 2)	4.000000	0.0000000
PROD(4, 3)	5.000000	0.0000000
PROD(4, 4)	6.000000	0.0000000
PROD(4, 5)	6.000000	0.0000000
PROD(4, 6)	6.000000	0.0000000
PROD(4, 7)	6.000000	0.0000000
PROD(5, 1)	4.000000	0.0000000
PROD(5, 2)	5.000000	0.0000000
PROD(5, 3)	6.000000	0.0000000
PROD(5, 4)	6.000000	0.0000000

PROD(5, 5)	6.000000	0.0000000
PROD(5, 6)	6.000000	0.0000000
PROD(5, 7)	6.000000	0.0000000
PROD(6, 1)	5.000000	0.0000000
PROD(6, 2)	6.000000	0.0000000
PROD(6, 3)	6.000000	0.0000000
PROD(6, 4)	6.000000	0.0000000
PROD(6, 5)	6.000000	0.0000000
PROD(6, 6)	6.000000	0.0000000
PROD(6, 7)	6.000000	0.0000000
PROD(7, 1)	6.000000	0.0000000
PROD(7, 2)	6.000000	0.0000000
PROD(7, 3)	6.000000	0.0000000
PROD(7, 4)	6.000000	0.0000000
PROD(7, 5)	6.000000	0.0000000
PROD(7, 6)	6.000000	0.0000000
PROD(7, 7)	6.000000	0.0000000
PROD(8, 1)	6.000000	0.0000000
PROD(8, 2)	6.000000	0.0000000
PROD(8, 3)	6.000000	0.0000000
PROD(8, 4)	6.000000	0.0000000
PROD(8, 5)	6.000000	0.0000000
PROD(8, 6)	6.000000	0.0000000
PROD(8, 7)	6.000000	0.0000000
PROD(9, 1)	6.000000	0.0000000
PROD(9, 2)	6.000000	0.0000000
PROD(9, 3)	6.000000	0.0000000
PROD(9, 4)	6.000000	0.0000000
PROD(9, 5)	6.000000	0.0000000
PROD(9, 6)	6.000000	0.0000000
PROD(9, 7)	6.000000	0.0000000
PROD(10, 1)	6.000000	0.0000000

PROD(10, 2)	6.000000	0.0000000
PROD(10, 3)	6.000000	0.0000000
PROD(10, 4)	6.000000	0.0000000
PROD(10, 5)	6.000000	0.0000000
PROD(10, 6)	6.000000	0.0000000
PROD(10, 7)	6.000000	0.0000000
PROD(11, 1)	6.000000	0.0000000
PROD(11, 2)	6.000000	0.0000000
PROD(11, 3)	6.000000	0.0000000
PROD(11, 4)	6.000000	0.0000000
PROD(11, 5)	6.000000	0.0000000
PROD(11, 6)	6.000000	0.0000000
PROD(11, 7)	6.000000	0.0000000
PROD(12, 1)	6.000000	0.0000000
PROD(12, 2)	6.000000	0.0000000
PROD(12, 3)	6.000000	0.0000000
PROD(12, 4)	6.000000	0.0000000
PROD(12, 5)	6.000000	0.0000000
PROD(12, 6)	6.000000	0.0000000
PROD(12, 7)	6.000000	0.0000000
PROD(13, 1)	6.000000	0.0000000
PROD(13, 2)	6.000000	0.0000000
PROD(13, 3)	6.000000	0.0000000
PROD(13, 4)	6.000000	0.0000000
PROD(13, 5)	6.000000	0.0000000
PROD(13, 6)	6.000000	0.0000000
PROD(13, 7)	6.000000	0.0000000
PROD(14, 1)	6.000000	0.0000000
PROD(14, 2)	6.000000	0.0000000
PROD(14, 3)	6.000000	0.0000000
PROD(14, 4)	6.000000	0.0000000
PROD(14, 5)	6.000000	0.0000000

PROD(14, 6)	6.000000	0.0000000
PROD(14, 7)	6.000000	0.0000000
PROD(15, 1)	6.000000	0.0000000
PROD(15, 2)	6.000000	0.0000000
PROD(15, 3)	6.000000	0.0000000
PROD(15, 4)	6.000000	0.0000000
PROD(15, 5)	6.000000	0.0000000
PROD(15, 6)	6.000000	0.0000000
PROD(15, 7)	6.000000	0.0000000
PROD(16, 1)	6.000000	0.0000000
PROD(16, 2)	6.000000	0.0000000
PROD(16, 3)	6.000000	0.0000000
PROD(16, 4)	6.000000	0.0000000
PROD(16, 5)	6.000000	0.0000000
PROD(16, 6)	6.000000	0.0000000
PROD(16, 7)	6.000000	0.0000000
PROD(17, 1)	6.000000	0.0000000
PROD(17, 2)	6.000000	0.0000000
PROD(17, 3)	6.000000	0.0000000
PROD(17, 4)	6.000000	0.0000000
PROD(17, 5)	6.000000	0.0000000
PROD(17, 6)	6.000000	0.0000000
PROD(17, 7)	6.000000	0.0000000
PROD(18, 1)	6.000000	0.0000000
PROD(18, 2)	6.000000	0.0000000
PROD(18, 3)	6.000000	0.0000000
PROD(18, 4)	6.000000	0.0000000
PROD(18, 5)	6.000000	0.0000000
PROD(18, 6)	6.000000	0.0000000
PROD(18, 7)	6.000000	0.0000000
PROD(19, 1)	6.000000	0.0000000
PROD(19, 2)	6.000000	0.0000000

PROD(19, 3)	6.000000	0.0000000
PROD(19, 4)	6.000000	0.0000000
PROD(19, 5)	6.000000	0.0000000
PROD(19, 6)	6.000000	0.0000000
PROD(19, 7)	6.000000	0.0000000
PROD(20, 1)	6.000000	0.0000000
PROD(20, 2)	6.000000	0.0000000
PROD(20, 3)	6.000000	0.0000000
PROD(20, 4)	6.000000	0.0000000
PROD(20, 5)	6.000000	0.0000000
PROD(20, 6)	6.000000	0.0000000
PROD(20, 7)	6.000000	0.0000000
PROD(21, 1)	6.000000	0.0000000
PROD(21, 2)	6.000000	0.0000000
PROD(21, 3)	6.000000	0.0000000
PROD(21, 4)	6.000000	0.0000000
PROD(21, 5)	6.000000	0.0000000
PROD(21, 6)	6.000000	0.0000000
PROD(21, 7)	6.000000	0.0000000

Row	Slack or Surplus	Dual Price
W_(1, 1)	3840.000	0.0000000
W_(1, 2)	3150.000	0.0000000
W_(1, 3)	2480.000	0.0000000
W_(1, 4)	1830.000	0.0000000
W_(1, 5)	1200.000	0.0000000
W_(1, 6)	590.0000	0.0000000
W_(1, 7)	0.0000000	-182.8234
W_(2, 1)	3250.000	0.0000000
W_(2, 2)	2560.000	0.0000000
W_(2, 3)	1890.000	0.0000000
W_(2, 4)	1240.000	0.0000000

W_(2, 5)	610.0000	0.0000000
W_(2, 6)	0.0000000	-1223.546
W_(2, 7)	47.72677	0.0000000
W_(3, 1)	2640.000	0.0000000
W_(3, 2)	1950.000	0.0000000
W_(3, 3)	1280.000	0.0000000
W_(3, 4)	630.0000	0.0000000
W_(3, 5)	0.0000000	-3650.813
W_(3, 6)	27.72677	0.0000000
W_(3, 7)	82.90508	0.0000000
W_(4, 1)	2010.000	0.0000000
W_(4, 2)	1320.000	0.0000000
W_(4, 3)	650.0000	0.0000000
W_(4, 4)	0.0000000	-6582.554
W_(4, 5)	7.726771	0.0000000
W_(4, 6)	42.90508	0.0000000
W_(4, 7)	102.6075	0.0000000
W_(5, 1)	1372.273	0.0000000
W_(5, 2)	682.2732	0.0000000
W_(5, 3)	12.27323	0.0000000
W_(5, 4)	0.0000000	-8424.958
W_(5, 5)	15.17831	0.0000000
W_(5, 6)	54.88078	0.0000000
W_(5, 7)	117.3239	0.0000000
W_(6, 1)	727.0949	0.0000000
W_(6, 2)	37.09492	0.0000000
W_(6, 3)	4.821690	0.0000000
W_(6, 4)	0.0000000	-8503.242
W_(6, 5)	19.70246	0.0000000
W_(6, 6)	62.14561	0.0000000
W_(6, 7)	126.4716	0.0000000
W_(7, 1)	77.39245	0.0000000

W_(7, 2)	25.11922	0.0000000
W_(7, 3)	0.2975350	0.0000000
W_(7, 4)	0.0000000	-6912.901
W_(7, 5)	22.44315	0.0000000
W_(7, 6)	66.76910	0.0000000
W_(7, 7)	132.3514	0.0000000
W_(8, 1)	65.11922	0.0000000
W_(8, 2)	20.29754	0.0000000
W_(8, 3)	0.0000000	-4228.009
W_(8, 4)	2.443147	0.0000000
W_(8, 5)	26.76910	0.0000000
W_(8, 6)	72.35144	0.0000000
W_(8, 7)	138.2579	0.0000000
W_(9, 1)	57.85439	0.0000000
W_(9, 2)	17.55685	0.0000000
W_(9, 3)	0.0000000	-1784.814
W_(9, 4)	4.325957	0.0000000
W_(9, 5)	29.90829	0.0000000
W_(9, 6)	75.81480	0.0000000
W_(9, 7)	141.2395	0.0000000
W_(10, 1)	53.23090	0.0000000
W_(10, 2)	15.67404	0.0000000
W_(10, 3)	0.0000000	-502.2392
W_(10, 4)	5.582333	0.0000000
W_(10, 5)	31.48884	0.0000000
W_(10, 6)	76.91352	0.0000000
W_(10, 7)	141.7422	0.0000000
W_(11, 1)	50.09171	0.0000000
W_(11, 2)	14.41767	0.0000000
W_(11, 3)	0.0000000	-135.2152
W_(11, 4)	5.906511	0.0000000
W_(11, 5)	31.33119	0.0000000

W_(11, 6)	76.15989	0.0000000
W_(11, 7)	140.7110	0.0000000
W_(12, 1)	52.00689	0.0000000
W_(12, 2)	17.58922	0.0000000
W_(12, 3)	3.495731	0.0000000
W_(12, 4)	8.920408	0.0000000
W_(12, 5)	33.74911	0.0000000
W_(12, 6)	78.30019	0.0000000
W_(13, 1)	55.17844	0.0000000
W_(13, 2)	21.08495	0.0000000
W_(13, 3)	6.509628	0.0000000
W_(13, 4)	11.33833	0.0000000
W_(13, 5)	35.88941	0.0000000
W_(14, 1)	58.67417	0.0000000
W_(14, 2)	24.09885	0.0000000
W_(14, 3)	8.927552	0.0000000
W_(14, 4)	13.47863	0.0000000
W_(15, 1)	61.68807	0.0000000
W_(15, 2)	26.51677	0.0000000
W_(15, 3)	11.06785	0.0000000
W_(16, 1)	64.10599	0.0000000
W_(16, 2)	28.65707	0.0000000
W_(17, 1)	66.24629	0.0000000