```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
OUTLINE OF THE MAIN PROCEDURES:

*  1. estimate model using first t obs
   2. simulate x_sim(t+tao) N times for tao-step ahead
prediction, based on parameters estimated step 1.
   3. calculate simulated conditional distribution
prob(u1<x_sim(t+tao)<u2|x(t))
   3. Repeat step 1,2,3 till t=R+P-tao.
   4. CS test
   5. Bootstrap Critical Value

                                                          *
****************************************************************
*********************************
                                                              *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
new; cls;
library co,pgraph;
#include co.ext
clearg xxt,xt,R,para1,para2,para3;//define global variables
see = 12343;
/*
xxt: is the data
model_ind: model indicator
    = 1 CIR
    = 2 SV
    = 3 SVJ
    = 4 CHEN
    = 5 CHEN_JUMP
    = 6 SM
*/

//output file = H:\lili\Norm\1989\CS_1989_L20.txt reset;
outwidth 255; output on; format /MA1 /LD 6,5;
load data[] =
D:\Research\continuous\code\Norm\Norm_Code\1989-1998\Rff1989_199
8.txt;//weekly data


xxt=data/100;//global variable stands for the whole sample



/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS out-of-sample specification test

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@ */
T=rows(xxt);
tao=1|2|3|4|5|6|12;  // steps of ahead simulation
```

```
S=100;//simulation times for x_sim(t+tao)
N=1;//Simulation for SV,

hh=1/2; // descretlize each interval
u_bar=(meanc(xxt)-0.5*stdc(xxt))|(meanc(xxt)+0.5*stdc(xxt));//th
e interval u_bar


//**************** the CS model selection test
*****************************

a=1/2;//define how many obs are in-sample obs, eg. a=1/2,  half
as in-sample, half out-of-sample,
R=round(a*T);


y1=time;

// estimate model based on model-indicator

// define the parameter estimation: global variables
para1={};//parameter estimation for CIR
para2={};//parameter estimation for SV
para3={};//parameter estimation for SVJ
P=100; // out-of-sample obs
R=T-P;//in-sample obs

/*
for RR(R,T-1,1);
    xt=xxt[1:RR,1];
    b1=estimate(xt,1);
    b2=estimate(xt,2);
    b3=estimate(xt,3);
    para1=para1|b1';
    para2=para2|b2';
    para3=para3|b3';
endfor;

" estimated parameters:";
para1; para2;para3;

*/

ppara1={0.729892  0.049231  0.091901
 0.672096  0.048181  0.087096
 0.694294  0.048541  0.088844
 0.619356  0.047584  0.082163
 0.735225  0.049515  0.091898
 0.697413  0.048704  0.088754
 0.667739  0.048307  0.086146
 0.651180  0.048378  0.084480
 0.723079  0.049482  0.090478
 0.677927  0.048749  0.086607
 0.699000  0.049095  0.088276
 0.662225  0.048670  0.085027
```

0.701353	0.049360	0.088251
0.693247	0.049200	0.087470
0.694889	0.049210	0.087511
0.704621	0.049273	0.088234
0.656536	0.048697	0.084040
0.718845	0.049645	0.089208
0.670282	0.048966	0.085024
0.721162	0.049654	0.089196
0.672705	0.048963	0.085051
0.703687	0.049470	0.087551
0.715098	0.049478	0.088403
0.687660	0.049035	0.086062
0.671796	0.048967	0.084601
0.671574	0.049218	0.084456
0.746423	0.050059	0.090554
0.656357	0.048806	0.083005
0.713582	0.049742	0.087692
0.692450	0.049420	0.085847
0.718694	0.049694	0.087912
0.681554	0.049182	0.084768
0.719058	0.049671	0.087751
0.685203	0.049187	0.084899
0.701452	0.049468	0.086130
0.706296	0.049487	0.086438
0.682210	0.049209	0.084369
0.698457	0.049522	0.085599
0.683811	0.049410	0.084293
0.716621	0.049832	0.086890
0.700404	0.049542	0.085484
0.703658	0.049563	0.085660
0.684309	0.049362	0.083985
0.716940	0.049782	0.086543
0.673427	0.049256	0.082905
0.714262	0.049860	0.086143
0.743589	0.049950	0.088381
0.599343	0.048180	0.076333
0.666496	0.049983	0.082074
0.806482	0.051219	0.093182
0.702667	0.049728	0.084774
0.717347	0.049944	0.085860
0.717405	0.049910	0.085774
0.725601	0.049936	0.086336
0.704599	0.049663	0.084590
0.711104	0.049800	0.085011
0.694858	0.049698	0.083616
0.739229	0.050205	0.087040
0.710067	0.049763	0.084673
0.727885	0.049937	0.085986
0.692632	0.049534	0.083126
0.713736	0.049908	0.084690
0.723520	0.049990	0.085373
0.721862	0.049898	0.085168
0.702600	0.049684	0.083576
0.709123	0.049849	0.083987
0.712375	0.049927	0.084153

```
 0.673305  0.049644  0.080930
 0.771675  0.050714  0.088561
 0.676519  0.049555  0.081051
 0.758941  0.050531  0.087400
 0.694389  0.049723  0.082326
 0.746712  0.050331  0.086289
 0.701604  0.049769  0.082740
 0.729229  0.050143  0.084784
 0.716295  0.049972  0.083710
 0.721171  0.050045  0.084000
 0.701815  0.049896  0.082408
 0.703188  0.050078  0.082421
 0.751216  0.050557  0.086040
 0.700991  0.049951  0.082094
 0.736992  0.050387  0.084786
 0.712662  0.050086  0.082839
 0.737179  0.050337  0.084631
 0.712927  0.050035  0.082706
 0.734054  0.050262  0.084233
 0.700276  0.049914  0.081570
 0.741045  0.050394  0.084592
 0.781041  0.050494  0.087483
 0.764443  0.049903  0.086440
 0.698961  0.048983  0.081848
 0.649099  0.048705  0.077609
 0.841732  0.050781  0.091487
 0.679516  0.048676  0.080293
 0.820738  0.050209  0.090148
 0.668816  0.048208  0.079871
 0.764546  0.049648  0.086196
 0.703589  0.048928  0.081856
 0.777448  0.049822  0.086903
 0.794201  0.049629  0.088387};

 ppara2={0.292902  0.039442  3.498784  0.000079  0.000000
-0.449918
 0.308071  0.040520  3.483204  0.000087  0.000000 -0.506733
 0.292885  0.039158  3.601142  0.000078  0.001694 -0.087370
 0.294868  0.039800  3.538742  0.000080  0.000000 -0.349653
 0.289098  0.039452  3.734903  0.000078  0.000000 -0.474006
 0.314255  0.041206  3.406030  0.000091  0.000000 -0.742521
 0.301248  0.040140  2.678307  0.000083  0.000000 -0.439839
 0.296380  0.040190  3.633664  0.000082  0.000000 -0.714177
 0.301804  0.040748  3.599381  0.000085  0.000000 -0.435238
 0.314773  0.041703  3.848854  0.000093  0.000000 -0.848954
 0.305935  0.040979  3.603798  0.000087  0.000000 -0.772342
 0.308240  0.041388  3.430982  0.000090  0.000000 -0.473037
 0.306171  0.041340  3.511999  0.000088  0.000000 -0.501110
 0.314428  0.041995  3.672991  0.000094  0.000000 -0.348713
 0.313140  0.041808  3.497731  0.000093  0.000000 -0.482403
 0.314133  0.041746  3.505930  0.000093  0.000000 -0.494112
 0.313478  0.041866  3.498815  0.000093  0.000000 -0.497156
 0.308929  0.041574  3.582637  0.000090  0.000000 -0.434360
 0.318016  0.042346  2.722727  0.000096  0.000000 -0.427160
 0.313026  0.041848  4.272069  0.000093  0.000000 -0.121538
```

0.319886	0.042367	3.540534	0.000097	0.000000	-0.450067
0.312399	0.041840	3.499383	0.000092	0.000000	-0.413046
0.319955	0.042203	3.879119	0.000096	0.000000	-0.303845
0.319348	0.042021	3.342129	0.000095	0.000000	-0.412847
0.312394	0.041696	3.488588	0.000091	0.000000	-0.406103
0.312473	0.042075	3.427361	0.000093	0.000000	-0.528740
0.317966	0.042360	3.276465	0.000096	0.003440	-0.071840
0.326648	0.042926	3.465773	0.000101	0.000000	-0.545819
0.312081	0.041957	3.693642	0.000092	0.026012	-0.242096
0.329288	0.043067	3.704663	0.000101	0.004237	-0.028321
0.320667	0.042431	3.474470	0.000096	0.024844	-0.264377
0.324452	0.042655	3.445882	0.000099	0.000000	-0.415075
0.319129	0.042198	3.504475	0.000095	0.000000	-0.594347
0.323799	0.042577	3.140810	0.000098	0.000000	-1.000000
0.318414	0.042160	3.965116	0.000095	0.027340	-0.109516
0.323498	0.042462	2.824531	0.000098	0.000000	-0.437444
0.322976	0.042483	3.293604	0.000098	0.000000	-0.453168
0.323180	0.042536	0.345557	0.000097	0.000000	-0.283750
0.324649	0.042852	3.814017	0.000099	0.000000	-0.581549
0.324709	0.042843	4.339784	0.000099	0.029330	-0.038860
0.330783	0.043144	3.120886	0.000103	0.000000	-0.504339
0.326792	0.042788	3.505688	0.000100	0.000000	-0.497154
0.326424	0.042835	3.214838	0.000100	0.000000	-0.412567
0.356977	0.044077	3.501892	0.000109	0.000000	-0.501559
0.325937	0.042961	3.497991	0.000101	0.000000	-0.510302
0.324241	0.042803	3.304932	0.000099	0.000000	-0.229246
0.330738	0.042888	3.496564	0.000103	0.000000	-0.484367
0.330155	0.043219	3.459565	0.000102	0.000000	-0.501563
0.298246	0.042132	2.274097	0.000089	0.000000	-0.346807
0.340109	0.044630	4.240789	0.000112	0.000000	-0.296413
0.348233	0.044638	3.607322	0.000115	0.000000	-0.535902
0.333109	0.043465	3.229029	0.000104	0.025882	-0.237755
0.337952	0.043705	3.613500	0.000107	0.000000	-0.486652
0.338056	0.043582	3.703643	0.000107	0.000000	-0.523415
0.337745	0.043527	4.187521	0.000106	0.000000	-0.406384
0.331929	0.043242	0.756485	0.000103	0.000023	-0.746649
0.334697	0.043542	1.831529	0.000105	0.007080	-0.000013
0.334876	0.043529	3.470228	0.000104	0.026857	-0.337441
0.343750	0.043930	3.879039	0.000110	0.000000	-0.395672
0.337457	0.043366	3.698332	0.000105	0.000000	-0.421098
0.339164	0.043577	3.645922	0.000106	0.000000	-0.380181
0.333212	0.043322	3.895775	0.000103	0.000000	-0.495595
0.341349	0.043802	5.764734	0.000108	0.000000	-0.372003
0.342862	0.043775	3.580801	0.000109	0.000000	-0.423320
0.338205	0.043545	3.444269	0.000105	0.026913	-0.363497
0.334586	0.043423	3.581263	0.000104	0.027244	-0.213607
0.341032	0.043847	3.554213	0.000108	0.000000	-0.607667
0.341386	0.044155	2.897344	0.000109	0.000000	-0.728039
0.343236	0.044155	4.940336	0.000110	0.000054	-0.524924
0.350660	0.044689	3.433288	0.000114	0.026496	-0.346098
0.336716	0.043719	2.779441	0.000106	0.000000	-0.506544
0.353526	0.044628	3.454366	0.000115	0.000000	-0.291826
0.341040	0.043837	3.433089	0.000107	0.027058	-0.194350
0.350577	0.044348	1.254052	0.000112	0.016775	-0.149921
0.341633	0.043838	5.028285	0.000107	0.004292	-0.370332

```
 0.354268  0.044453  3.492566  0.000114  0.000000 -0.503404
 0.344862  0.044029  3.494939  0.000109  0.010762 -0.451393
 0.346788  0.044241  4.157548  0.000111  0.000000 -0.449374
 0.342856  0.044267  3.524443  0.000108  0.027617 -0.412335
 0.351177  0.044662  3.236637  0.000113  0.027095 -0.139056
 0.357215  0.044950  3.308472  0.000116  0.027729 -0.282938
 0.347825  0.044409  3.707574  0.000111  0.011189 -0.345006
 0.356160  0.044839  3.044664  0.000116  0.000000 -0.664610
 0.351844  0.044528  3.465160  0.000113  0.000000 -0.723266
 0.351479  0.044538  3.584625  0.000113  0.028437 -0.328351
 0.353819  0.044591  3.237251  0.000114  0.000000 -0.933937
 0.354011  0.044627  3.521846  0.000114  0.000365 -0.301588
 0.346623  0.044267  4.101044  0.000111  0.000000 -0.646657
 0.362300  0.044583  4.724308  0.000118  0.000000 -0.887570
 0.359142  0.043817  3.502144  0.000113  0.028120 -0.410351
 0.336459  0.042370  3.364702  0.000100  0.023489 -0.436708
 0.309311  0.041111  3.495900  0.000087  0.024718 -0.414049
 0.318899  0.041614  2.440468  0.000092  0.021103 -0.373280
 0.361096  0.043797  3.383659  0.000114  0.000000 -0.275522
 0.300489  0.039621  3.555437  0.000082  0.016745 -0.349720
 0.337173  0.042089  3.515947  0.000099  0.010635 -0.471930
 0.275168  0.037647  2.765075  0.000070  0.019660 -0.280938
 0.316395  0.041002  3.251888  0.000089  0.023691 -0.461947
 0.297166  0.039700  3.546410  0.000080  0.023772 -0.453703
 0.330204  0.041300  3.532386  0.000095  0.000913 -0.483119 };

ppara3={
 0.267537  0.046919  2.292001  0.000083  0.007328 -0.104420
5.497900  0.001100
3.121000  0.002000
 0.282578  0.047032  2.309636  0.000084  0.009031 -0.089103
5.497900  0.001100
3.121000  0.002000
 0.262704  0.046570  2.292381  0.000082  0.007313 -0.107812
5.497900  0.001100
3.121000  0.002000
 0.268827  0.046901  2.294707  0.000083  0.009007 -0.120563
5.497900  0.001100
3.121000  0.002000
 0.239062  0.046582  2.292172  0.000080  0.009598 -0.105067
5.497900  0.001100
3.121000  0.002000
 0.268494  0.046990  2.292573  0.000083  0.008517 -0.118975
5.497900  0.001100
3.121000  0.002000
 0.270647  0.046922  2.292220  0.000082  0.006332 -0.105882
5.497900  0.001100
3.121000  0.002000
 0.245279  0.046919  2.292097  0.000081  0.008207 -0.102729
5.497900  0.001100
3.121000  0.002000
 0.262880  0.047442  2.292183  0.000082  0.009131 -0.106058
5.497900  0.001100
3.121000  0.002000
 0.257865  0.047315  2.292136  0.000082  0.010344 -0.105049
```

```
5.497900  0.001100
3.121000  0.002000
 0.264725  0.047381  2.292025  0.000082  0.009908 -0.098050
5.497900  0.001100
3.121000  0.002000
 0.289542  0.047998  2.182708  0.000084  0.011591 -0.097275
5.497900  0.001100
3.121000  0.002000
 0.284485  0.048023  2.292308  0.000083  0.011856 -0.106256
5.497900  0.001100
3.121000  0.002000
 0.275648  0.047883  2.291909  0.000083  0.009348 -0.109982
5.497900  0.001100
3.121000  0.002000
 0.284097  0.047929  2.292348  0.000083  0.011560 -0.115642
5.497900  0.001100
3.121000  0.002000
 0.277930  0.047663  2.285363  0.000082  0.010513 -0.100181
5.497900  0.001100
3.121000  0.002000
 0.282047  0.047913  2.292674  0.000082  0.008734 -0.100276
5.497900  0.001100
3.121000  0.002000
 0.278339  0.047851  2.292660  0.000082  0.004336 -0.121024
5.497900  0.001100
3.121000  0.002000
 0.265814  0.047764  2.292235  0.000081  0.008310 -0.110043
5.497900  0.001100
3.121000  0.002000
 0.264331  0.047583  2.291934  0.000080  0.010203 -0.107264
5.497900  0.001100
3.121000  0.002000
 0.281607  0.047872  2.291977  0.000081  0.007358 -0.106397
5.497900  0.001100
3.121000  0.002000
 0.274009  0.047719  2.292172  0.000080  0.007599 -0.104284
5.497900  0.001100
3.121000  0.002000
 0.284423  0.047701  2.230448  0.000081  0.002002 -0.094829
5.497900  0.001100
3.121000  0.002000
 0.287354  0.047600  2.160255  0.000081  0.009231 -0.091348
5.497900  0.001100
3.121000  0.002000
 0.280884  0.047634  2.292689  0.000080  0.000882 -0.113301
5.497900  0.001100
3.121000  0.002000
 0.272668  0.047931  2.292692  0.000080  0.007822 -0.110414
5.497900  0.001100
3.121000  0.002000
 0.279965  0.047894  2.292152  0.000081  0.009776 -0.105240
5.497900  0.001100
3.121000  0.002000
 0.274259  0.047873  2.292046  0.000080  0.009198 -0.101879
5.497900  0.001100
```

```
3.121000  0.002000
 0.254802  0.047490  2.292228  0.000078  0.009984 -0.105720
5.497900  0.001100
3.121000  0.002000
 0.265387  0.047725  2.292070  0.000079  0.008152 -0.103960
5.497900  0.001100
3.121000  0.002000
 0.277237  0.047668  2.292135  0.000079  0.010267 -0.104598
5.497900  0.001100
3.121000  0.002000
 0.287009  0.047766  2.308291  0.000079  0.011258 -0.105484
5.497900  0.001100
3.121000  0.002000
 0.277155  0.047360  2.292389  0.000079  0.007050 -0.104719
5.497900  0.001100
3.121000  0.002000
 0.271530  0.047422  2.291601  0.000078  0.005211 -0.124471
5.497900  0.001100
3.121000  0.002000
 0.286029  0.047653  2.290321  0.000078  0.009144 -0.105248
5.497900  0.001100
3.121000  0.002000
 0.269076  0.047287  2.292056  0.000077  0.005663 -0.106778
5.497900  0.001100
3.121000  0.002000
 0.280731  0.047582  2.292222  0.000078  0.007080 -0.105899
5.497900  0.001100
3.121000  0.002000
 0.292654  0.047835  2.318379  0.000078  0.013813 -0.110223
5.497900  0.001100
3.121000  0.002000
 0.282507  0.047867  2.292405  0.000078  0.010732 -0.103782
5.497900  0.001100
3.121000  0.002000
 0.269299  0.047642  2.292127  0.000077  0.007213 -0.105314
5.497900  0.001100
3.121000  0.002000
 0.278486  0.047733  2.291957  0.000077  0.006112 -0.102651
5.497900  0.001100
3.121000  0.002000
 0.286642  0.047653  2.414953  0.000077  0.011654 -0.073712
5.497900  0.001100
3.121000  0.002000
 0.280115  0.047616  2.292281  0.000077  0.007394 -0.103732
5.497900  0.001100
3.121000  0.002000
 0.275176  0.047474  2.292089  0.000076  0.008741 -0.104247
5.497900  0.001100
3.121000  0.002000
 0.278406  0.047662  2.292121  0.000076  0.007302 -0.105806
5.497900  0.001100
3.121000  0.002000
 0.279478  0.047683  2.292462  0.000076  0.010837 -0.090159
5.497900  0.001100
3.121000  0.002000
```

```
 0.281119  0.047322  2.292031  0.000077  0.008401 -0.113567
5.497900  0.001100
3.121000  0.002000
 0.294699  0.047975  2.304232  0.000078  0.008917 -0.090997
5.497900  0.001100
3.121000  0.002000
 0.285505  0.048792  2.292209  0.000080  0.011098 -0.106959
5.497900  0.001100
3.121000  0.002000
 0.288634  0.049002  2.310416  0.000083  0.009305 -0.103151
5.497900  0.001100
3.121000  0.002000
 0.289644  0.048591  2.292158  0.000083  0.007124 -0.104869
5.497900  0.001100
3.121000  0.002000
 0.294169  0.048333  2.291977  0.000082  0.002139 -0.118708
5.497900  0.001100
3.121000  0.002000
 0.302958  0.048416  2.383401  0.000083  0.008263 -0.119199
5.497900  0.001100
3.121000  0.002000
 0.295559  0.048204  2.292109  0.000082  0.010011 -0.095001
5.497900  0.001100
3.121000  0.002000
 0.290097  0.048029  2.290063  0.000081  0.008997 -0.079031
5.497900  0.001100
3.121000  0.002000
 0.292105  0.048068  2.292200  0.000081  0.004831 -0.102327
5.497900  0.001100
3.121000  0.002000
 0.287734  0.048192  2.292223  0.000081  0.010279 -0.107607
5.497900  0.001100
3.121000  0.002000
 0.299289  0.048353  2.128511  0.000082  0.012174 -0.085120
5.497900  0.001100
3.121000  0.002000
 0.293146  0.048098  2.292201  0.000081  0.003741 -0.105276
5.497900  0.001100
3.121000  0.002000
 0.294378  0.047919  2.290284  0.000081  0.011227 -0.084188
5.497900  0.001100
3.121000  0.002000
 0.241371  0.046667  2.292153  0.000077  0.001371 -0.100216
5.497900  0.001100
3.121000  0.002000
 0.286939  0.047969  2.292450  0.000080  0.008713 -0.105581
5.497900  0.001100
3.121000  0.002000
 0.295554  0.048108  2.291061  0.000080  0.006646 -0.118500
5.497900  0.001100
3.121000  0.002000
 0.292323  0.047906  2.296132  0.000080  0.004115 -0.111501
5.497900  0.001100
3.121000  0.002000
 0.288174  0.047834  2.292210  0.000079  0.001752 -0.099012
```

```
5.497900  0.001100
3.121000  0.002000
 0.305715  0.048164  2.292967  0.000080  0.004546 -0.147921
5.497900  0.001100
3.121000  0.002000
 0.290927  0.048047  2.292428  0.000079  0.006095 -0.105543
5.497900  0.001100
3.121000  0.002000
 0.294261  0.048459  2.696940  0.000079  0.009686 -0.089377
5.497900  0.001100
3.121000  0.002000
 0.288787  0.048277  2.292134  0.000079  0.008816 -0.105646
5.497900  0.001100
3.121000  0.002000
 0.284919  0.048307  2.292129  0.000078  0.008871 -0.104775
5.497900  0.001100
3.121000  0.002000
 0.291468  0.048163  2.292012  0.000078  0.007840 -0.113349
5.497900  0.001100
3.121000  0.002000
 0.281261  0.048081  2.292197  0.000077  0.008509 -0.105160
5.497900  0.001100
3.121000  0.002000
 0.281335  0.047881  2.292162  0.000077  0.007591 -0.106972
5.497900  0.001100
3.121000  0.002000
 0.291456  0.048063  2.293598  0.000077  0.008410 -0.110086
5.497900  0.001100
3.121000  0.002000
 0.291751  0.048007  2.302464  0.000077  0.018501 -0.038308
5.497900  0.001100
3.121000  0.002000
 0.285083  0.047858  2.292318  0.000076  0.013027 -0.098803
5.497900  0.001100
3.121000  0.002000
 0.283661  0.047979  2.291845  0.000076  0.003623 -0.105292
5.497900  0.001100
3.121000  0.002000
 0.286404  0.048059  2.292651  0.000076  0.002563 -0.112228
5.497900  0.001100
3.121000  0.002000
 0.296496  0.048456  2.293072  0.000076  0.009499 -0.113919
5.497900  0.001100
3.121000  0.002000
 0.271088  0.048035  2.292170  0.000075  0.009728 -0.108299
5.497900  0.001100
3.121000  0.002000
 0.280782  0.048221  2.292289  0.000075  0.009553 -0.106571
5.497900  0.001100
3.121000  0.002000
 0.278791  0.048070  2.292009  0.000075  0.007729 -0.103701
5.497900  0.001100
3.121000  0.002000
 0.286574  0.048203  2.292066  0.000075  0.011375 -0.100243
5.497900  0.001100
```

```
3.121000  0.002000
 0.290499  0.048118  2.292136  0.000075  0.006237 -0.111069
5.497900  0.001100
3.121000  0.002000
 0.285575  0.048021  2.291929  0.000074  0.008662 -0.102987
5.497900  0.001100
3.121000  0.002000
 0.286079  0.047898  2.291711  0.000074  0.007673 -0.104216
5.497900  0.001100
3.121000  0.002000
 0.287514  0.048039  2.292476  0.000074  0.007321 -0.109348
5.497900  0.001100
3.121000  0.002000
 0.284917  0.047950  2.292090  0.000074  0.008489 -0.104898
5.497900  0.001100
3.121000  0.002000
 0.285008  0.047416  2.292472  0.000075  0.011320 -0.106477
5.497900  0.001100
3.121000  0.002000
 0.284437  0.046532  2.292526  0.000077  0.008987 -0.108589
5.497900  0.001100
3.121000  0.002000
 0.286225  0.046235  2.292203  0.000076  0.010344 -0.107302
5.497900  0.001100
3.121000  0.002000
 0.302377  0.046976  2.273720  0.000079  0.018185 -0.010507
5.497900  0.001100
3.121000  0.002000
 0.293910  0.046664  2.292269  0.000078  0.004235 -0.111490
5.497900  0.001100
3.121000  0.002000
 0.289473  0.046622  2.283106  0.000077  0.012183 -0.092588
5.497900  0.001100
3.121000  0.002000
 0.273046  0.045532  2.292139  0.000076  0.007086 -0.105498
5.497900  0.001100
3.121000  0.002000
 0.290643  0.046188  1.967238  0.000076  0.011086 -0.155747
5.497900  0.001100
3.121000  0.002000
 0.275229  0.045629  2.292038  0.000075  0.004777 -0.111339
5.497900  0.001100
3.121000  0.002000
 0.280222  0.046244  2.292180  0.000075  0.010359 -0.101065
5.497900  0.001100
3.121000  0.002000
 0.280328  0.046053  2.292140  0.000075  0.006219 -0.111168
5.497900  0.001100
3.121000  0.002000
 0.288618  0.045741  2.281831  0.000077  0.010323 -0.103034
5.497900  0.001100
3.121000  0.002000
};

for i(1,100,1);
```

```
para1=para1|ppara1[3*i-2:i*3];
para2=para2|ppara2[6*i-5:i*6];
para3=para3|ppara3[10*i-9:i*10];
endfor;


/* */
// CS model selection test statistic
mod_ind1=1;//CIR as the benchmark
v={};//hold vp~v1~v2;
vp={};// D(k,P,N), hold test statistic for all 2 alternative
models
v1={};// D1(k,P,N),
v2={};// D2(k,P,N),
V_sup={};// sup(D(k,P,N))

// simulate the latent variables ONLY ONCE
//{svSim1,mvSim1,svSim2,mvSim2}=latentSim(mod_ind1,2,N,hh,R,R);
// using the first R obs
{svSim1,mvSim1,svSim2,mvSim2}=latentSim(2,2,N,hh,R,R); // using
the first R obs

SimInd=3;//  simulation indicator: 1= Con_den(),2=Con_den2()
means simulate the latent variables ONLY ONCE ,
         // 3=Con_den4(); dont simulate latent variables, using
v_bar



/*  */
if SimInd==1;
    v= v|CS_stat(xxt,R,N,tao,S,mod_ind1,1,hh,u_bar);  //Against
CIR
    v= v|CS_stat(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar);  //Against
SVJ
elseif SimInd==2;
    v=
v|CS_stat2(xxt,R,N,tao,S,mod_ind1,1,hh,u_bar,svSim1,mvSim1,svSim
2,mvSim2); //against CIR
    v=
v|CS_stat2(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar,svSim1,mvSim1,svSim
2,mvSim2);  //Against SVJ
elseif SimInd==3;
    v= v|CS_stat3(xxt,R,N,tao,S,mod_ind1,2,hh,u_bar); //against
SV
    v= v|CS_stat3(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar);  //Against
SVJ
endif;

print " The CS stat D=D1-D2 for tao=1,2,3,4,5, 6, 12 are:" v;
for i(1,rows(tao),1);
    if v[i,1]>=v[i+rows(tao),1];
        V_sup=V_sup|v[i,1];
        v1=v1|v[i,2];
        v2=v2|v[i,3];
```

```
    else;
        V_sup=V_sup|v[i+rows(tao),1];
        v1=v1|v[i+rows(tao),2];
        v2=v2|v[i+rows(tao),3];
    endif;
endfor;



print "The CS V_sup~v1~v2 for tao=1,2,3,4,5, 6, 12 are: "
V_sup~v1~v2;



// doing b times to get the  Bootstrap critical value for CS

option=1; //using the same parameters as from X(t)
//option=2; // using different parameters in Bootstrap

mod_ind1=1;//CIR as the benchmark
lval=5; //block size
V_sup={};
SimInd=3;//dont simulate latent


for boot(1,100,1);    //bootstrap replications
    xxt_boot= boot1(xxt,lval,see);
    v_boot={};
    if SimInd==1;
// skip this becoz it is tedious to simulate a long series for
SV each time each step

    elseif SimInd==2;
        v_boot=
v_boot~(CS_statBoot2(xxt_boot,R,N,tao,S,mod_ind1,1,hh,u_bar,svSi
m1,mvSim1,svSim2,mvSim2,option)); //CIR,  recenter,P=(1-a)*T, so
sqrt(P/T)=sqrt(1-a)

        v_boot=
v_boot~(CS_statBoot2(xxt_boot,R,N,tao,S,mod_ind1,3,hh,u_bar,svSi
m1,mvSim1,svSim2,mvSim2,option));//SVJ

   elseif SimInd==3;
        v_boot=
v_boot~(CS_statBoot3(xxt_boot,R,N,tao,S,mod_ind1,2,hh,u_bar,opti
on)); //SV,  recenter,P=(1-a)*T, so sqrt(P/T)=sqrt(1-a)

        v_boot=
v_boot~(CS_statBoot3(xxt_boot,R,N,tao,S,mod_ind1,3,hh,u_bar,opti
on));//SVJ

    endif;

    V_sup=V_sup|(maxc(v_boot'))';
    print "V_sup for tao=1,2,4,12 are: " V_sup;
```

```
    print "boot=" boot;
    see=see+100;// change see to change the rndu in boot
endfor;

print " the bootsrap CS stat are: " V_sup;
print " the 5%, 10%,15%,20% critical values for tao=1,2 4,12 are
: " ;


for loop(1,7,1);
    V_b=sortc(V_sup,loop);
    v_b[95,loop]~v_b[90,loop]~v_b[85,loop]~v_b[80,loop];
endfor;


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$

" for u_bar=(meanc(xxt)-1*stdc(xxt))|(meanc(xxt)+1*stdc(xxt))";
//the interval u_bar

see=12343;
u_bar=(meanc(xxt)-1*stdc(xxt))|(meanc(xxt)+1*stdc(xxt));//the
interval u_bar

// CS model selection test statistic
// CS model selection test statistic
mod_ind1=1;//CIR as the benchmark
v={};//hold vp~v1~v2;
vp={};// D(k,P,N), hold test statistic for all 2 alternative
models
v1={};// D1(k,P,N),
v2={};// D2(k,P,N),
V_sup={};// sup(D(k,P,N))



/*  */
if SimInd==1;
    v= v|CS_stat(xxt,R,N,tao,S,mod_ind1,1,hh,u_bar);  //Against
CIR
    v= v|CS_stat(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar);  //Against
SVJ
elseif SimInd==2;
    v=
v|CS_stat2(xxt,R,N,tao,S,mod_ind1,1,hh,u_bar,svSim1,mvSim1,svSim
2,mvSim2); //against CIR
    v=
v|CS_stat2(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar,svSim1,mvSim1,svSim
2,mvSim2);  //Against SVJ
elseif SimInd==3;
    v= v|CS_stat3(xxt,R,N,tao,S,mod_ind1,2,hh,u_bar); //against
SV
```

```
    v= v|CS_stat3(xxt,R,N,tao,S,mod_ind1,3,hh,u_bar);  //Against
SVJ
endif;

print " The CS stat D=D1-D2 for tao=1,2,3,4,5, 6, 12 are:" v;
for i(1,rows(tao),1);
    if v[i,1]>=v[i+rows(tao),1];
        V_sup=V_sup|v[i,1];
        v1=v1|v[i,2];
        v2=v2|v[i,3];
    else;
        V_sup=V_sup|v[i+rows(tao),1];
        v1=v1|v[i+rows(tao),2];
        v2=v2|v[i+rows(tao),3];
    endif;
endfor;



print "The CS V_sup~v1~v2 for tao=1,2,3,4,5, 6, 12 are: "
V_sup~v1~v2;



// doing b times to get the  Bootstrap critical value for CS

option=1; //using the same parameters as from X(t)
//option=2; // using different parameters in Bootstrap

mod_ind1=1;//CIR as the benchmark
lval=5; //block size
V_sup={};
SimInd=3;//dont simulate latent


for boot(1,100,1);    //bootstrap replications
    xxt_boot= boot1(xxt,lval,see);
    v_boot={};
    if SimInd==1;
// skip this becoz it is tedious to simulate a long series for
SV each time each step

    elseif SimInd==2;
        v_boot=
v_boot~(CS_statBoot2(xxt_boot,R,N,tao,S,mod_ind1,1,hh,u_bar,svSi
m1,mvSim1,svSim2,mvSim2,option)); //CIR,  recenter,P=(1-a)*T, so
sqrt(P/T)=sqrt(1-a)

        v_boot=
v_boot~(CS_statBoot2(xxt_boot,R,N,tao,S,mod_ind1,3,hh,u_bar,svSi
m1,mvSim1,svSim2,mvSim2,option));//SVJ

   elseif SimInd==3;
        v_boot=
v_boot~(CS_statBoot3(xxt_boot,R,N,tao,S,mod_ind1,2,hh,u_bar,opti
```

```
on)); //SV,  recenter,P=(1-a)*T, so sqrt(P/T)=sqrt(1-a)

        v_boot=
v_boot~(CS_statBoot3(xxt_boot,R,N,tao,S,mod_ind1,3,hh,u_bar,opti
on));//SVJ

    endif;

    V_sup=V_sup|(maxc(v_boot'))';
    print "V_sup for tao=1,2,4,12 are: " V_sup;

    print "boot=" boot;
    see=see+100;// change see to change the rndu in boot
endfor;

print " the bootsrap CS stat are: " V_sup;
print " the 5%, 10%,15%,20% critical values for tao=1,2 4,12 are
: " ;


for loop(1,7,1);
    V_b=sortc(V_sup,loop);
    v_b[95,loop]~v_b[90,loop]~v_b[85,loop]~v_b[80,loop];
endfor;


y2=time;
print "time is :" y2-y1;
end;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   Lden: returns the Kernal density
                                *
****************************************************************
*********************************
* Inputs:         arg    -   the values at which the density is
to evaluated                    *
*             v      -   The realised values of the series for
which density is to be estimates *
* Output:     p      -   density
                                *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc  Lden(arg,v);
local Narg,r,N,h,P,j,d,t1;
    Narg = rows(arg);
    r=1/(4+cols(v));
    N=rows(v);
    h=(1/(N^r))*stdc(v)';
    P=zeros(Narg,1);
    j=0;
    do while j < Narg;j=j+1;
        d=(arg[j,.]-V)./h;
```

```
        t1=pdfn(d)./h;
        t1=prodc(t1');
        P[j]=meanc(t1);
    endo;
retp(P);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_CJ: Returns the obj func for CHEN-JUMP Model
                               *
****************************************************************
*********************************
* Inputs:       b      -   starting values
                                   *
* Output:             -   the objective function to be minimised
                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_CJ(b);
   local
moms,g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,gsubT,gsubTall,NWlags,NW,inv
NW,flagnw,g11;
        moms=CJ_moms(b);
   g1=meanc(xt[2:rows(xt)-3]-moms[1:rows(moms)-4,1]);
   g2=meanc(xt[2:rows(xt)-3]^2-moms[1:rows(moms)-4,2]);
   g3=meanc(xt[2:rows(xt)-3]^3-moms[1:rows(moms)-4,3]);
   g4=meanc(xt[2:rows(xt)-3]^4-moms[1:rows(moms)-4,4]);

g5=meanc(xt[2:rows(xt)-3].*xt[3:rows(xt)-2]-moms[1:rows(moms)-4,
5]);

g6=meanc(xt[2:rows(xt)-3].*xt[4:rows(xt)-1]-moms[1:rows(moms)-4,
6]);

g7=meanc(xt[2:rows(xt)-3].*xt[5:rows(xt)]-moms[1:rows(moms)-4,7]
);

g8=meanc((xt[2:rows(xt)-3]^2).*xt[3:rows(xt)-2]-moms[1:rows(moms
)-4,8]);

g9=meanc((xt[2:rows(xt)-3]^2).*xt[4:rows(xt)-1]-moms[1:rows(moms
)-4,9]);

g10=meanc(xt[2:rows(xt)-3].*(xt[3:rows(xt)-2]^2)-moms[1:rows(mom
s)-4,10]);

g11=meanc(xt[2:rows(xt)-3].*(xt[4:rows(xt)-1]^2)-moms[1:rows(mom
s)-4,11]);
   gsubT=g1|g2|g3|g4|g5|g6|g7|g8|g9|g10|g11;
   gsubTall=((xt[2:rows(xt)-3]-moms[1:rows(moms)-4,1]) ~
(xt[2:rows(xt)-3]^2-moms[1:rows(moms)-4,2])
            ~(xt[2:rows(xt)-3]^3-moms[1:rows(moms)-4,3]) ~
(xt[2:rows(xt)-3]^4-moms[1:rows(moms)-4,4])
```

```
~(xt[2:rows(xt)-3].*xt[3:rows(xt)-2]-moms[1:rows(moms)-4,5]) ~
(xt[2:rows(xt)-3].*xt[4:rows(xt)-1]-moms[1:rows(moms)-4,6])

~(xt[2:rows(xt)-3].*xt[5:rows(xt)]-moms[1:rows(moms)-4,7]) ~
((xt[2:rows(xt)-3]^2).*xt[3:rows(xt)-2]-moms[1:rows(moms)-4,8])

~((xt[2:rows(xt)-3]^2).*xt[4:rows(xt)-1]-moms[1:rows(moms)-4,9])
~
(xt[2:rows(xt)-3].*(xt[3:rows(xt)-2]^2)-moms[1:rows(moms)-4,10])


~(xt[2:rows(xt)-3].*(xt[4:rows(xt)-1]^2)-moms[1:rows(moms)-4,11]
)   );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;
proc CJ_moms(b);
local moms,m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11;

m1=(exp((-((b[1]+b[10]+b[8]))).*(1/52)).*((exp(b[1]*(1/52))))^((
b[10]+b[8])/b[1]).*((b[1]*xt-(((-1)+exp(b[1]*(1/52)))).*((b[11]*
b[10]-b[9]*b[8]))+b[1]^2*b[6]*(1/52))))/b[1];

m2=(1/(b[1]^2*b[5]^3)).*((exp((-((2*b[1]+b[5]+b[10]+b[8]))).*(1/
52)).*((exp(b[1]*(1/52))))^((b[10]+b[8])/b[1]).*((exp(((2*b[1]+b
[5])).*(1/52))*b[5]^3*((((b[11]*b[10]-b[9]*b[8]))^2+b[1]*((b[11]
^2*b[10]+b[9]^2*b[8])))))+b[1]^4*b[6]*b[7]^2-2*exp(((b[1]+b[5])).
*(1/52))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).*((b[1]*xt+b[11]*b[10]
-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(b[5]*(1/52)).*((b[5]^3*((b[1
1]*b[10]-b[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11]^2*b[
10]-2*xt*b[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(1/52)+
b[1]^2*b[5]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])
).*(1/52)))+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5
]*(1/52))))))))));

m3=(1/(2*b[1]^3*b[5]^6)).*((exp((-((3*b[1]+3*b[5]+b[10]+b[8]))).
*(1/52)).*((exp(b[1]*(1/52))))^((b[10]+b[8])/b[1]).*(((-2)*exp(3
*b[5]*(1/52)).*(((-1)+exp(b[1]*(1/52))))^3*b[5]^6*((b[11]*b[10]-
b[9]*b[8]))^3-6*exp(3*b[5]*(1/52)).*(((-1)+exp(b[1]*(1/52))))^2*
b[1]*b[5]^6*((b[11]*b[10]-b[9]*b[8])).*((xt*(((-b[11])*b[10]+b[9
]*b[8]))+((1+exp(b[1]*(1/52)))).*((b[11]^2*b[10]+b[9]^2*b[8])))))
+2*exp(3*b[5]*(1/52))*b[1]^3*b[5]^6*((xt^3+3*(((-1)+exp(2*b[1]*(
1/52)))).*((b[11]^2*b[10]+b[9]^2*b[8]))*b[6]*(1/52)+3*xt*((b[3]-
2*(((-1)+exp(b[1]*(1/52)))).*((b[11]*b[10]-b[9]*b[8]))*b[6])).*(
1/52)))+6*exp(2*b[5]*(1/52))*b[1]^5*b[5]^3*xt*b[6]*((exp(b[5]*(1
/52))*b[5]^3*b[6]*(1/52)^2+b[7]^2*((1+exp(b[5]*(1/52)).*(((-1)+b
```

```
[5]*(1/52)))))))+2*exp(2*b[5]*(1/52))*b[1]^6*b[5]*b[6]*((exp(b[5
]*(1/52))*b[5]^5*b[6]^2*(1/52)^3+3*b[7]^4*((2+b[5]*(1/52)+exp(b[
5]*(1/52)).*(((-2)+b[5]*(1/52)))))+3*b[5]^2*b[6]*b[7]^2*(1/52).*
((1+exp(b[5]*(1/52)).*(((-1)+b[5]*(1/52)))))))-3*b[1]^4*b[6]*(((
-(((-1)+exp(b[1]*(1/52))))).*(((-1)+exp(b[5]*(1/52))))^2*((1+exp
(b[5]*(1/52)))).*((b[5]^2))^(3/2).*((b[11]*b[10]-b[9]*b[8]))*b[7
]^2+(((-1)+exp(b[1]*(1/52)))).*(((-1)+exp(3*b[5]*(1/52))))*b[5]^
4*((b[11]*b[10]-b[9]*b[8]))*b[7]^2*(1/52)-(((-1)+exp(3*b[5]*(1/5
2))))*b[5]^6*(1/52).*((xt^2+((b[3]-(((-1)+exp(b[1]*(1/52)))).*((
b[11]*b[10]-b[9]*b[8]))*b[6])).*(1/52)))-((1+exp(3*b[5]*(1/52)))
)*b[5]^6*(1/52).*((xt^2+((b[3]-(((-1)+exp(b[1]*(1/52)))).*((b[11
]*b[10]-b[9]*b[8]))*b[6])).*(1/52)))+(((-1)+exp(b[1]*(1/52))))*b
[5]^3*((b[11]*b[10]-b[9]*b[8]))*b[7]^2*((1-exp(b[5]*(1/52))+exp(
2*b[5]*(1/52))+b[5]*(1/52)+exp(3*b[5]*(1/52)).*(((-1)+b[5]*(1/52
)))))))+2*exp(3*b[5]*(1/52))*b[1]^2*b[5]^6*(((-3).*(((-1)+exp(b[
1]*(1/52))))*xt^2*((b[11]*b[10]-b[9]*b[8]))+3*(((-1)+exp(2*b[1]*
(1/52))))*xt*((b[11]^2*b[10]+b[9]^2*b[8]))-(((-1)+exp(b[1]*(1/52
))))).*((2*((1+exp(b[1]*(1/52))+exp(2*b[1]*(1/52))))*b[11]^3*b[10
]-3*(((-1)+exp(b[1]*(1/52))))*b[11]^2*b[10]^2*b[6]*(1/52)+3*b[11
]*b[10]*((b[3]+2*(((-1)+exp(b[1]*(1/52))))*b[9]*b[8]*b[6])).*(1/
52)-b[9]*b[8]*((2*((1+exp(b[1]*(1/52))+exp(2*b[1]*(1/52))))*b[9]
^2+3*b[3]*(1/52)+3*(((-1)+exp(b[1]*(1/52))))*b[9]*b[8]*b[6]*(1/5
2))))))))));

m4=(1/b[1]^4).*((exp((-((4*b[1]+b[10]+b[8]))).*(1/52)).*((exp(b[
1]*(1/52))))^((b[10]+b[8])/b[1]).*((((b[1]*xt-(((-1)+exp(b[1]*(1
/52)))).*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*b[6]*(1/52)))^4+(1/b[5
]^5).*((4*exp((-b[5]).*(1/52))*b[1]^2*((b[1]*xt-(((-1)+exp(b[1]*
(1/52))))).*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*b[6]*(1/52))).*(((-2
)*exp(((3*b[1]+b[5])).*(1/52))*b[5]^5*((b[11]^3*b[10]-b[9]^3*b[8
]))+3*b[1]^4*b[6]*b[7]^4*((2+b[5]*(1/52)))+exp(b[5]*(1/52)).*((2
*b[5]^5*((b[11]^3*b[10]-b[9]^3*b[8]))-6*b[1]^4*b[6]*b[7]^4+3*b[1
]^4*b[5]*b[6]*b[7]^4*(1/52)))))))+(1/b[5]^3).*((6*exp((-b[5]).*(
1/52))*b[1]*((b[1]*xt-(((-1)+exp(b[1]*(1/52)))).*((b[11]*b[10]-b
[9]*b[8]))+b[1]^2*b[6]*(1/52)))^2*((exp(((2*b[1]+b[5])).*(1/52))
*b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]))+b[1]^3*b[6]*b[7]^2-exp(b[5
]*(1/52)).*((b[1]^3*b[6]*b[7]^2-b[1]^3*b[5]*b[6]*b[7]^2*(1/52)+b
[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]-b[1]*b[3]*(1/52)))))))))+(1/b[
5]^6).*((3*exp((-2)*b[5]*(1/52))*b[1]^2*((exp(((2*b[1]+b[5])).*(
1/52))*b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]))+b[1]^3*b[6]*b[7]^2-e
xp(b[5]*(1/52)).*((b[1]^3*b[6]*b[7]^2-b[1]^3*b[5]*b[6]*b[7]^2*(1
/52)+b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]-b[1]*b[3]*(1/52)))))))^2
))+(1/(2*b[2]^3*b[5]^7)).*((3*exp((-((b[2]+2*b[5]))).*(1/52))*b[
1]^3*((4*exp(((4*b[1]+b[2]+2*b[5])).*(1/52))*b[2]^3*b[5]^7*((b[1
1]^4*b[10]+b[9]^4*b[8]))+2*exp(2*b[5]*(1/52))*b[1]*b[5]^7*b[3]*b
[4]^2+exp(b[2]*(1/52))*b[1]^5*b[2]^3*b[6]*b[7]^6+4*exp(((b[2]+b[
5])).*(1/52))*b[1]^5*b[2]^3*b[6]*b[7]^6*((7+5*b[5]*(1/52)+b[5]^2
*(1/52)^2))-exp(((b[2]+2*b[5])).*(1/52)).*((2*b[1]*b[5]^7*b[3]*b
[4]^2-2*b[1]*b[2]*b[5]^7*b[3]*b[4]^2*(1/52)+b[2]^3*((4*b[5]^7*((
b[11]^4*b[10]+b[9]^4*b[8]))+29*b[1]^5*b[6]*b[7]^6-10*b[1]^5*b[5]
*b[6]*b[7]^6*(1/52)))))))))));

m5=((-((1/(b[1]^2*b[5]^3)).*((exp((-2)*b[1]*(1/52)-b[5]*(1/52)-3
*b[1]*(1/52)-2*b[5]*(1/52)).*(((-exp(2*b[1]*(1/52)+b[5]*(1/52)+3
*b[1]*(1/52)+2*b[5]*(1/52)))*b[5]^3*((b[11]*b[10]-b[9]*b[8]))^2+
```

```
exp(b[5]*(((1/52)+2*(1/52)))+b[1]*(((1/52)+3*(1/52))))*b[1]*b[5]
^3*(((-b[11]^2)*b[10]+b[1]*(1/52)*b[11]*b[10]*b[6]-b[9]*b[8]*((b
[9]+b[1]*(1/52)*b[6])))-exp(((b[1]+b[5])).*(((1/52)+(1/52))))*b
[1]^4*b[6]*b[7]^2+exp(2*b[1]*(((1/52)+(1/52)))+b[5]*(((1/52)+2*(
1/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).*((b[1]*xt+b[11]*b[10]
-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(((b[1]+b[5])).*(((1/52)+2*(1
/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8]-b[1]^2*(1/52)*b[6])).*((b
[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))-exp(b[1]*(((1/
52)+(1/52)))+b[5]*(((1/52)+2*(1/52)))).*((b[5]^3*((b[11]*b[10]-b
[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11]^2*b[10]-2*xt*b
[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(1/52)+b[1]^2*b[5
]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])).*(1/52))
)+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))
))))))))))));

m6=((-((1/(b[1]^2*b[5]^3)).*((exp((-2)*b[1]*(2/52)-b[5]*(2/52)-3
*b[1]*(1/52)-2*b[5]*(1/52)).*(((-exp(2*b[1]*(2/52)+b[5]*(2/52)+3
*b[1]*(1/52)+2*b[5]*(1/52)))*b[5]^3*((b[11]*b[10]-b[9]*b[8]))^2+
exp(b[5]*(((2/52)+2*(1/52)))+b[1]*(((2/52)+3*(1/52))))*b[1]*b[5]
^3*(((-b[11]^2)*b[10]+b[1]*(2/52)*b[11]*b[10]*b[6]-b[9]*b[8]*((b
[9]+b[1]*(2/52)*b[6])))-exp(((b[1]+b[5])).*(((2/52)+(1/52))))*b
[1]^4*b[6]*b[7]^2+exp(2*b[1]*(((2/52)+(1/52)))+b[5]*(((2/52)+2*(
1/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).*((b[1]*xt+b[11]*b[10]
-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(((b[1]+b[5])).*(((2/52)+2*(1
/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8]-b[1]^2*(2/52)*b[6])).*((b
[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))-exp(b[1]*(((2/
52)+(1/52)))+b[5]*(((2/52)+2*(1/52)))).*((b[5]^3*((b[11]*b[10]-b
[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11]^2*b[10]-2*xt*b
[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(1/52)+b[1]^2*b[5
]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])).*(1/52))
)+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))
))))))))))));

m7=((-((1/(b[1]^2*b[5]^3)).*((exp((-2)*b[1]*(3/52)-b[5]*(3/52)-3
*b[1]*(1/52)-2*b[5]*(1/52)).*(((-exp(2*b[1]*(3/52)+b[5]*(3/52)+3
*b[1]*(1/52)+2*b[5]*(1/52)))*b[5]^3*((b[11]*b[10]-b[9]*b[8]))^2+
exp(b[5]*(((3/52)+2*(1/52)))+b[1]*(((3/52)+3*(1/52))))*b[1]*b[5]
^3*(((-b[11]^2)*b[10]+b[1]*(3/52)*b[11]*b[10]*b[6]-b[9]*b[8]*((b
[9]+b[1]*(3/52)*b[6])))-exp(((b[1]+b[5])).*(((3/52)+(1/52))))*b
[1]^4*b[6]*b[7]^2+exp(2*b[1]*(((3/52)+(1/52)))+b[5]*(((3/52)+2*(
1/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).*((b[1]*xt+b[11]*b[10]
-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(((b[1]+b[5])).*(((3/52)+2*(1
/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8]-b[1]^2*(3/52)*b[6])).*((b
[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))-exp(b[1]*(((3/
52)+(1/52)))+b[5]*(((3/52)+2*(1/52)))).*((b[5]^3*((b[11]*b[10]-b
[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11]^2*b[10]-2*xt*b
[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(1/52)+b[1]^2*b[5
]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])).*(1/52))
)+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))
))))))))))));

m8=((-((1/(b[1]^3*b[5]^5)).*((exp((-3)*b[1]*(1/52)-b[5]*(1/52)-5
*b[1]*(1/52)-3*b[5]*(1/52)).*((exp(3*b[1]*(1/52)+b[5]*(1/52)+5*b
[1]*(1/52)+3*b[5]*(1/52))*b[5]^5*((b[11]*b[10]-b[9]*b[8])).*((((
b[11]*b[10]-b[9]*b[8]))^2+b[1]*((b[11]^2*b[10]+b[9]^2*b[8])))))-e
```

xp(2*b[1]*(1/52)+b[5]*(1/52)+5*b[1]*(1/52)+3*b[5]*(1/52))*b[1]*b
[5]^5*(((-2)*b[11]^3*b[10]^2+2*b[11]^2*b[9]*b[10]*b[8]-2*b[11]*b
[9]^2*b[10]*b[8]+2*b[9]^3*b[8]^2+b[1]^2*(1/52).*((b[11]^2*b[10]+
b[9]^2*b[8]))*b[6]+b[1]*(((-2)*b[11]^3*b[10]+(1/52)*b[11]^2*b[10
]^2*b[6]-2*(1/52)*b[11]*b[9]*b[10]*b[8]*b[6]+b[9]^2*b[8]*((2*b[9
]+(1/52)*b[8]*b[6])))))+exp(3*b[1]*(((1/52)+(1/52)))+b[5]*(((1/
52)+2*(1/52))))*b[1]^4*b[5]^2*((b[11]*b[10]-b[9]*b[8]))*b[6]*b[7
]^2-exp(2*b[1]*(1/52)+b[5]*(1/52)+3*b[1]*(1/52)+2*b[5]*(1/52))*b
[1]^4*b[5]^2*b[6]*(((-2)*b[11]*b[10]+2*b[9]*b[8]+b[1]^2*(1/52)*b
[6]))*b[7]^2-2*exp(3*b[1]*(1/52)+b[5]*(1/52)+4*b[1]*(1/52)+3*b[5
]*(1/52))*b[5]^5*((b[11]*b[10]-b[9]*b[8]))^2*((b[1]*xt+b[11]*b[1
0]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(2*b[1]*(((1/52)+2*(1/52)))
+b[5]*(((1/52)+3*(1/52))))*b[5]^5*(((-((b[11]*b[10]-b[9]*b[8]))^
2)-3*b[1]*((b[11]^2*b[10]+b[9]^2*b[8]))+2*b[1]^2*(1/52).*((b[11]
*b[10]-b[9]*b[8]))*b[6])).*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^
2*b[6]*(1/52)))+exp(3*b[1]*(((1/52)+(1/52)))+b[5]*(((1/52)+3*(1/
52))))*b[5]*((b[11]*b[10]-b[9]*b[8])).*((b[5]^4*((b[11]*b[10]-b[
9]*b[8]))^2+b[1]*b[5]^4*((2*xt*b[11]*b[10]-b[11]^2*b[10]-2*xt*b[
9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^4*xt*b[6]*(1/52)+b[1]^2*b[5]
^4*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])).*(1/52)))
+b[1]^4*b[5]*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/5
2)))))))+exp(2*b[1]*(1/52)+b[5]*(1/52)+3*b[1]*(1/52)+3*b[5]*(1/5
2))*b[5]*((2*b[11]*b[10]-2*b[9]*b[8]-b[1]^2*(1/52)*b[6])).*((b[5
]^4*((b[11]*b[10]-b[9]*b[8]))^2+b[1]*b[5]^4*((2*xt*b[11]*b[10]-b
[11]^2*b[10]-2*xt*b[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^4*xt*b[6
]*(1/52)+b[1]^2*b[5]^4*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b
[8]*b[6])).*(1/52)))+b[1]^4*b[5]*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7
]^2*(((-1)+b[5]*(1/52)))))))-3*exp(2*b[1]*(((1/52)+(1/52)))+b[5]
(((1/52)+2(1/52))))*b[1]^4*b[6]*b[7]^2*((b[1]*b[5]^2*xt+b[5]^2
*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*((b[5]^2*b[6]*(1/52)+b[7]^2*((
2+b[5]*(1/52)))))))-exp(2*b[1]*(((1/52)+(1/52)))+b[5]*(((1/52)+3
(1/52)))).((b[5]^5*((b[11]*b[10]-b[9]*b[8]))^3*-3*b[1]*b[5]^5*
((b[11]*b[10]-b[9]*b[8])).*(((-xt)*b[11]*b[10]+b[11]^2*b[10]+xt*
b[9]*b[8]+b[9]^2*b[8]))+b[1]^3*b[5]^5*((xt^3-3*((b[11]^2*b[10]+b
[9]^2*b[8]))*b[6]*(1/52)+3*xt*((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b
[8]*b[6])).*(1/52)))+3*b[1]^5*b[5]^2*xt*b[6]*((b[5]^3*b[6]*(1/52
)^2+b[7]^2*(((-1)+b[5]*(1/52)))))+b[1]^6*b[6]*((b[5]^5*b[6]^2*(1
/52)^3+3*b[7]^4*(((-2)+b[5]*(1/52)))+3*b[5]^2*b[6]*b[7]^2*(1/52)
.*(((-1)+b[5]*(1/52)))))+b[1]^2*b[5]^5*((2*b[11]^3*b[10]+3*xt^2*
((b[11]*b[10]-b[9]*b[8]))-3*xt*((b[11]^2*b[10]+b[9]^2*b[8]))+3*b
[11]^2*b[10]^2*b[6]*(1/52)+3*b[11]*b[10]*((b[3]-2*b[9]*b[8]*b[6]
)).*(1/52)+b[9]*b[8]*(((-2)*b[9]^2-3*b[3]*(1/52)+3*b[9]*b[8]*b[6
]*(1/52)))))+3*b[1]^4*b[5]^2*b[6]*((b[5]^3*(1/52).*((xt^2+b[3]*(
1/52)))+b[11]*b[10]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1
/52)))))-b[9]*b[8]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/
52))))))))))))));

m9=((-((1/(b[1]^3*b[5]^5)).*((exp((-3)*b[1]*(2/52)-b[5]*(2/52)-5
b[1](1/52)-3*b[5]*(1/52)).*((exp(3*b[1]*(2/52)+b[5]*(2/52)+5*b
[1]*(1/52)+3*b[5]*(1/52))*b[5]^5*((b[11]*b[10]-b[9]*b[8])).*((((
b[11]*b[10]-b[9]*b[8]))^2+b[1]*((b[11]^2*b[10]+b[9]^2*b[8])))))-e
xp(2*b[1]*(2/52)+b[5]*(2/52)+5*b[1]*(1/52)+3*b[5]*(1/52))*b[1]*b
[5]^5*(((-2)*b[11]^3*b[10]^2+2*b[11]^2*b[9]*b[10]*b[8]-2*b[11]*b
[9]^2*b[10]*b[8]+2*b[9]^3*b[8]^2+b[1]^2*(2/52).*((b[11]^2*b[10]+
b[9]^2*b[8]))*b[6]+b[1]*(((-2)*b[11]^3*b[10]+(2/52)*b[11]^2*b[10

```
]^2*b[6]-2*(2/52)*b[11]*b[9]*b[10]*b[8]*b[6]+b[9]^2*b[8]*((2*b[9
]+(2/52)*b[8]*b[6])))))+exp(3*b[1]*(((2/52)+(1/52)))+b[5]*(((2/
52)+2*(1/52))))*b[1]^4*b[5]^2*((b[11]*b[10]-b[9]*b[8]))*b[6]*b[7
]^2-exp(2*b[1]*(2/52)+b[5]*(2/52)+3*b[1]*(1/52)+2*b[5]*(1/52))*b
[1]^4*b[5]^2*b[6]*(((-2)*b[11]*b[10]+2*b[9]*b[8]+b[1]^2*(2/52)*b
[6]))*b[7]^2-2*exp(3*b[1]*(2/52)+b[5]*(2/52)+4*b[1]*(1/52)+3*b[5
]*(1/52))*b[5]^5*((b[11]*b[10]-b[9]*b[8]))^2*((b[1]*xt+b[11]*b[1
0]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+exp(2*b[1]*(((2/52)+2*(1/52)))
+b[5]*(((2/52)+3*(1/52))))*b[5]^5*(((-((b[11]*b[10]-b[9]*b[8]))^
2)-3*b[1]*((b[11]^2*b[10]+b[9]^2*b[8]))+2*b[1]^2*(2/52).*((b[11]
*b[10]-b[9]*b[8]))*b[6])).*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^
2*b[6]*(1/52)))+exp(3*b[1]*(((2/52)+(1/52)))+b[5]*(((2/52)+3*(1/
52))))*b[5]*((b[11]*b[10]-b[9]*b[8])).*((b[5]^4*((b[11]*b[10]-b[
9]*b[8]))^2+b[1]*b[5]^4*((2*xt*b[11]*b[10]-b[11]^2*b[10]-2*xt*b[
9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^4*xt*b[6]*(1/52)+b[1]^2*b[5]
^4*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]*b[6])).*(1/52)))
+b[1]^4*b[5]*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/5
2)))))))+exp(2*b[1]*(2/52)+b[5]*(2/52)+3*b[1]*(1/52)+3*b[5]*(1/5
2))*b[5]*((2*b[11]*b[10]-2*b[9]*b[8]-b[1]^2*(2/52)*b[6])).*((b[5
]^4*((b[11]*b[10]-b[9]*b[8]))^2+b[1]*b[5]^4*((2*xt*b[11]*b[10]-b
[11]^2*b[10]-2*xt*b[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^4*xt*b[6
]*(1/52)+b[1]^2*b[5]^4*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b
[8]*b[6])).*(1/52)))+b[1]^4*b[5]*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7
]^2*(((-1)+b[5]*(1/52)))))))-3*exp(2*b[1]*(((2/52)+(1/52)))+b[5]
*(((2/52)+2*(1/52))))*b[1]^4*b[6]*b[7]^2*((b[1]*b[5]^2*xt+b[5]^2
*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*((b[5]^2*b[6]*(1/52)+b[7]^2*((
2+b[5]*(1/52)))))))-exp(2*b[1]*(((2/52)+(1/52)))+b[5]*(((2/52)+3
*(1/52)))).*((b[5]^5*((b[11]*b[10]-b[9]*b[8]))^3*-3*b[1]*b[5]^5*
((b[11]*b[10]-b[9]*b[8])).*(((-xt)*b[11]*b[10]+b[11]^2*b[10]+xt*
b[9]*b[8]+b[9]^2*b[8]))+b[1]^3*b[5]^5*((xt^3-3*((b[11]^2*b[10]+b
[9]^2*b[8]))*b[6]*(1/52)+3*xt*((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b
[8]*b[6])).*(1/52)))+3*b[1]^5*b[5]^2*xt*b[6]*((b[5]^3*b[6]*(1/52
)^2+b[7]^2*(((-1)+b[5]*(1/52)))))+b[1]^6*b[6]*((b[5]^5*b[6]^2*(1
/52)^3+3*b[7]^4*(((-2)+b[5]*(1/52)))+3*b[5]^2*b[6]*b[7]^2*(1/52)
.*(((-1)+b[5]*(1/52)))))+b[1]^2*b[5]^5*((2*b[11]^3*b[10]+3*xt^2*
((b[11]*b[10]-b[9]*b[8]))-3*xt*((b[11]^2*b[10]+b[9]^2*b[8]))+3*b
[11]^2*b[10]^2*b[6]*(1/52)+3*b[11]*b[10]*((b[3]-2*b[9]*b[8]*b[6]
)).*(1/52)+b[9]*b[8]*(((-2)*b[9]^2-3*b[3]*(1/52)+3*b[9]*b[8]*b[6
]*(1/52)))))+3*b[1]^4*b[5]^2*b[6]*((b[5]^3*(1/52).*((xt^2+b[3]*(
1/52)))+b[11]*b[10]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1
/52)))))-b[9]*b[8]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/
52))))))))))))))));

m10=((1/(2*b[1]^2*b[5]^5)).*((2*exp((-2)*b[1]*(1/52)-3*b[1]*(1/5
2)-b[5]*(1/52))*b[1]*(((-2)*exp(((3*b[1]+b[5])).*(1/52))*b[5]^5*
((b[11]^3*b[10]-b[9]^3*b[8]))+3*b[1]^4*b[6]*b[7]^4*((2+b[5]*(1/5
2)))+exp(b[5]*(1/52)).*((2*b[5]^5*((b[11]^3*b[10]-b[9]^3*b[8]))-
6*b[1]^4*b[6]*b[7]^4+3*b[1]^4*b[5]*b[6]*b[7]^4*(1/52)))))+4*exp(
(-2)*b[1]*(1/52)-3*b[1]*(1/52)-b[5]*(1/52))*b[5]^2*((b[1]*xt-(((
-1)+exp(b[1]*(((1/52)+(1/52))))))).*((b[11]*b[10]-b[9]*b[8]))+b[1
]^2*b[6]*((exp(b[1]*(1/52)).*(1/52)+(1/52))))).*((exp(((2*b[1]+b
[5])).*(1/52))*b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]))+b[1]^3*b[6]*
b[7]^2-exp(b[5]*(1/52)).*((b[1]^3*b[6]*b[7]^2-b[1]^3*b[5]*b[6]*b
[7]^2*(1/52)+b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]-b[1]*b[3]*(1/52)
))))))+(1/b[1]).*((2*exp((-2).*((b[5]*(((1/52)+(1/52)))+b[1]*((3
```

```
*(1/52)+2*(1/52))))))*b[5]^2*((b[1]*xt-(((-1)+exp(b[1]*(1/52))))
.*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*b[6]*(1/52))).*((exp(2*b[5]*(
((1/52)+(1/52)))+3*b[1]*((2*(1/52)+(1/52))))*b[5]^3*((((b[11]*b[
10]-b[9]*b[8]))^2+b[1]*((b[11]^2*b[10]+b[9]^2*b[8])))))-2*exp(2*b
[5]*(((1/52)+(1/52)))+b[1]*((5*(1/52)+3*(1/52))))*b[1]^2*b[5]^3*
(1/52).*((b[11]*b[10]-b[9]*b[8]))*b[6]+exp(4*b[1]*(1/52)+2*b[5]*
(1/52)+b[1]*(1/52)+b[5]*(1/52))*b[1]^4*b[6]*b[7]^2+exp(4*b[1]*(1
/52)+b[5]*(1/52)+3*b[1]*(1/52)+2*b[5]*(1/52))*b[1]^4*b[6]*b[7]^2
+exp(2*b[5]*(((1/52)+(1/52)))+b[1]*((4*(1/52)+3*(1/52))))*b[1]^2
*((b[5]^3*(1/52).*((b[3]+b[1]^2*(1/52)*b[6]^2))-b[1]^2*b[6]*b[7]
^2+b[1]^2*b[5]*(1/52)*b[6]*b[7]^2))-2*exp(2*b[5]*(((1/52)+(1/52)
))+b[1]*((5*(1/52)+2*(1/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).
*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+2*exp(2*b[
5]*(((1/52)+(1/52)))+2*b[1]*((2*(1/52)+(1/52))))*b[1]^2*b[5]^3*(
1/52)*b[6]*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+
exp(2*b[5]*(((1/52)+(1/52)))+b[1]*((4*(1/52)+(1/52)))).*((b[5]^3
*((b[11]*b[10]-b[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11
]^2*b[10]-2*xt*b[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(
1/52)+b[1]^2*b[5]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]
*b[6])).*(1/52)))+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-
1)+b[5]*(1/52))))))))))))));

m11=((1/(2*b[1]^2*b[5]^5)).*((2*exp((-2)*b[1]*(2/52)-3*b[1]*(1/5
2)-b[5]*(1/52))*b[1]*(((-2)*exp(((3*b[1]+b[5])).*(1/52))*b[5]^5*
((b[11]^3*b[10]-b[9]^3*b[8]))+3*b[1]^4*b[6]*b[7]^4*((2+b[5]*(1/5
2)))+exp(b[5]*(1/52)).*((2*b[5]^5*((b[11]^3*b[10]-b[9]^3*b[8]))-
6*b[1]^4*b[6]*b[7]^4+3*b[1]^4*b[5]*b[6]*b[7]^4*(1/52)))))+4*exp(
(-2)*b[1]*(2/52)-3*b[1]*(1/52)-b[5]*(1/52))*b[5]^2*((b[1]*xt-(((
-1)+exp(b[1]*(((2/52)+(1/52))))))).*((b[11]*b[10]-b[9]*b[8]))+b[1
]^2*b[6]*((exp(b[1]*(1/52)).*(2/52)+(1/52))))).*((exp(((2*b[1]+b
[5])).*(1/52))*b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]))+b[1]^3*b[6]*
b[7]^2-exp(b[5]*(1/52)).*((b[1]^3*b[6]*b[7]^2-b[1]^3*b[5]*b[6]*b
[7]^2*(1/52)+b[5]^3*((b[11]^2*b[10]+b[9]^2*b[8]-b[1]*b[3]*(1/52)
))))))+(1/b[1]).*((2*exp((-2).*((b[5]*(((2/52)+(1/52)))+b[1]*((3
*(2/52)+2*(1/52))))))*b[5]^2*((b[1]*xt-(((-1)+exp(b[1]*(1/52))))
.*((b[11]*b[10]-b[9]*b[8]))+b[1]^2*b[6]*(1/52))).*((exp(2*b[5]*(
((2/52)+(1/52)))+3*b[1]*((2*(2/52)+(1/52))))*b[5]^3*((((b[11]*b[
10]-b[9]*b[8]))^2+b[1]*((b[11]^2*b[10]+b[9]^2*b[8])))))-2*exp(2*b
[5]*(((2/52)+(1/52)))+b[1]*((5*(2/52)+3*(1/52))))*b[1]^2*b[5]^3*
(2/52).*((b[11]*b[10]-b[9]*b[8]))*b[6]+exp(4*b[1]*(2/52)+2*b[5]*
(2/52)+b[1]*(1/52)+b[5]*(1/52))*b[1]^4*b[6]*b[7]^2+exp(4*b[1]*(2
/52)+b[5]*(2/52)+3*b[1]*(1/52)+2*b[5]*(1/52))*b[1]^4*b[6]*b[7]^2
+exp(2*b[5]*(((2/52)+(1/52)))+b[1]*((4*(2/52)+3*(1/52))))*b[1]^2
*((b[5]^3*(2/52).*((b[3]+b[1]^2*(2/52)*b[6]^2))-b[1]^2*b[6]*b[7]
^2+b[1]^2*b[5]*(2/52)*b[6]*b[7]^2))-2*exp(2*b[5]*(((2/52)+(1/52)
))+b[1]*((5*(2/52)+2*(1/52))))*b[5]^3*((b[11]*b[10]-b[9]*b[8])).
*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+2*exp(2*b[
5]*(((2/52)+(1/52)))+2*b[1]*((2*(2/52)+(1/52))))*b[1]^2*b[5]^3*(
2/52)*b[6]*((b[1]*xt+b[11]*b[10]-b[9]*b[8]+b[1]^2*b[6]*(1/52)))+
exp(2*b[5]*(((2/52)+(1/52)))+b[1]*((4*(2/52)+(1/52)))).*((b[5]^3
*((b[11]*b[10]-b[9]*b[8]))^2+b[1]*b[5]^3*((2*xt*b[11]*b[10]-b[11
]^2*b[10]-2*xt*b[9]*b[8]-b[9]^2*b[8]))+2*b[1]^3*b[5]^3*xt*b[6]*(
1/52)+b[1]^2*b[5]^3*((xt^2+((b[3]+2*b[11]*b[10]*b[6]-2*b[9]*b[8]
*b[6])).*(1/52)))+b[1]^4*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-
1)+b[5]*(1/52))))))))))))));
```

```
 moms=m1~m2~m3~m4~m5~m6~m7~m8~m9~m10~m11;
retp(moms);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_C: Returns the obj func for CHEN Model
                                 *
****************************************************************
*********************************
* Inputs:       b      -   starting values
                                    *
* Output:           -   the objective function to be minimised
                                *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_C(b);
   local
moms,g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,gsubT,gsubTall,NWlags,NW,inv
NW,flagnw,g11,g12,g13,g14;
        moms=C_moms(b);
   g1=meanc(xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]);
   g2=meanc(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2]);
   g3=meanc(xt[2:rows(xt)-1]^3-moms[1:rows(moms)-2,3]);
   g4=meanc(xt[2:rows(xt)-1]^4-moms[1:rows(moms)-2,4]);
   g5=meanc(xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,5]);
   g6=meanc(xt[2:rows(xt)-1].*(xt[3:rows(xt)]^2)
-moms[1:rows(moms)-2,6]);
   g7=meanc((xt[2:rows(xt)-1]^2).*(xt[3:rows(xt)])
-moms[1:rows(moms)-2,7]);
   gsubT=g1|g2|g3|g4|g5|g6|g7;
   gsubTall=((xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]) ~
(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2])
            ~(xt[2:rows(xt)-1]^3-moms[1:rows(moms)-2,3]) ~
(xt[2:rows(xt)-1]^4-moms[1:rows(moms)-2,4])
            ~(xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,5])
            ~(xt[2:rows(xt)-1].*(xt[3:rows(xt)]^2)
-moms[1:rows(moms)-2,6])
            ~((xt[2:rows(xt)-1]^2).*(xt[3:rows(xt)])
-moms[1:rows(moms)-2,7])   );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;
proc C_moms(b);
```

```
local moms,m1,m2,m3,m4,m5,m6,m7;
    m1=exp((-b[1]).*(1/52)).*((xt+b[1]*b[6]*(1/52)));

m2=(exp((-((2*b[1]+b[5]))).*(1/52)).*((b[1]^2*b[6]*b[7]^2+exp(b[
5]*(1/52)).*(((-b[1]^2)*b[6]*b[7]^2+b[1]^2*b[5]*b[6]*b[7]^2*(1/5
2)+b[5]^3*((xt^2+2*b[1]*xt*b[6]*(1/52)+(1/52).*((b[3]+b[1]^2*b[6
]^2*(1/52)))))))))/b[5]^3;

m3=(1/b[5]^5).*((exp((-((3*b[1]+b[5]))).*(1/52)).*((3*b[1]^2*b[6
]*b[7]^2*((2*b[1]*b[7]^2+b[1]*b[5]*b[7]^2*(1/52)+b[5]^2*((xt+b[1
]*b[6]*(1/52)))))+exp(b[5]*(1/52)).*(((-6)*b[1]^3*b[6]*b[7]^4+3*
b[1]^3*b[5]*b[6]*b[7]^4*(1/52)-3*b[1]^2*b[5]^2*b[6]*b[7]^2*((xt+
b[1]*b[6]*(1/52)))+3*b[1]^2*b[5]^3*b[6]*b[7]^2*(1/52).*((xt+b[1]
*b[6]*(1/52)))+b[5]^5*((xt+b[1]*b[6]*(1/52))).*((xt^2+3*b[3]*(1/
52)+2*b[1]*xt*b[6]*(1/52)+b[1]^2*b[6]^2*(1/52)^2)))))));

m4=(1/(2*b[2]^3*b[5]^7)).*((exp((-((4*b[1]+b[2]+2*b[5]))).*(1/52
)).*((6*exp(2*b[5]*(1/52))*b[5]^7*b[3]*b[4]^2+3*exp(b[2]*(1/52))
*b[1]^4*b[2]^3*b[6]*b[7]^4*((2*b[5]*b[6]+b[7]^2))+12*exp(((b[2]+
b[5])).*(1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2*((7*b[1]^2*b[7]^4+2*b[
1]*b[5]^3*b[7]^2*(1/52).*((xt+b[1]*b[6]*(1/52)))+b[1]^2*b[5]*b[7
]^2*(((-b[6])+5*b[7]^2*(1/52)))+b[5]^4*((xt^2+2*b[1]*xt*b[6]*(1/
52)+(1/52).*((b[3]+b[1]^2*b[6]^2*(1/52)))))+b[1]*b[5]^2*b[7]^2*(
(4*xt+b[1]*(1/52).*((5*b[6]+b[7]^2*(1/52)))))))+exp(((b[2]+2*b[5
])).*(1/52)).*(((-6)*b[5]^7*b[3]*b[4]^2+6*b[2]*b[5]^7*b[3]*b[4]^
2*(1/52)+b[2]^3*(((-87)*b[1]^4*b[6]*b[7]^6-12*b[1]^3*b[5]^2*b[6]
*b[7]^4*((4*xt+5*b[1]*b[6]*(1/52)))+6*b[1]^3*b[5]^3*b[6]*b[7]^4*
(1/52).*((4*xt+5*b[1]*b[6]*(1/52)))+6*b[1]^4*b[5]*b[6]*b[7]^4*((
b[6]+5*b[7]^2*(1/52)))-12*b[1]^2*b[5]^4*b[6]*b[7]^2*((xt^2+2*b[1
]*xt*b[6]*(1/52)+(1/52).*((b[3]+b[1]^2*b[6]^2*(1/52)))))+12*b[1]
^2*b[5]^5*b[6]*b[7]^2*(1/52).*((xt^2+2*b[1]*xt*b[6]*(1/52)+(1/52
).*((b[3]+b[1]^2*b[6]^2*(1/52)))))+2*b[5]^7*((xt^4+4*b[1]*xt^3*b
[6]*(1/52)+6*xt^2*(1/52).*((b[3]+b[1]^2*b[6]^2*(1/52)))+4*b[1]*x
t*b[6]*(1/52)^2*((3*b[3]+b[1]^2*b[6]^2*(1/52)))+(1/52)^2*((3*b[3
]^2+6*b[1]^2*b[3]*b[6]^2*(1/52)+b[1]^4*b[6]^4*(1/52)^2)))))))))
));

m5=(((1/b[5]^3).*((exp((-b[1]).*(1/52)-2*b[1]*(1/52)-b[5]*(1/52)
).*((b[1]^2*b[6]*b[7]^2+exp(((b[1]+b[5])).*(1/52))*b[1]*b[5]^3*(
1/52)*b[6]*((xt+b[1]*b[6]*(1/52)))+exp(b[5]*(1/52)).*(((-b[1]^2)
*b[6]*b[7]^2+b[1]^2*b[5]*b[6]*b[7]^2*(1/52)+b[5]^3*((xt^2+2*b[1]
*xt*b[6]*(1/52)+(1/52).*((b[3]+b[1]^2*b[6]^2*(1/52)))))))))))));

m6=(((1/b[5]^5).*((exp((-4)*b[1]*(1/52)-b[5]*(1/52)-6*b[1]*(1/52
)-3*b[5]*(1/52)).*((exp(3*b[1]*(1/52)+b[5]*(1/52)+4*b[1]*(1/52)+
2*b[5]*(1/52))*b[1]^3*b[5]^2*(1/52)*b[6]^2*b[7]^2+3*exp(3*b[1]*(
((2/52)))+b[5]*(((1/52)+2*(1/52))))*b[1]^2*b[6]*b[7]^2*((2*b[1]*
b[7]^2+b[1]*b[5]*b[7]^2*(1/52)+b[5]^2*((xt+b[1]*b[6]*(1/52)))))+
exp(3*b[1]*(1/52)+b[5]*(1/52)+4*b[1]*(1/52)+3*b[5]*(1/52))*b[1]*
b[5]^2*(1/52)*b[6]*(((-b[1]^2)*b[6]*b[7]^2+b[1]^2*b[5]*b[6]*b[7]
^2*(1/52)+b[5]^3*((xt^2+2*b[1]*xt*b[6]*(1/52)+(1/52).*((b[3]+b[1
]^2*b[6]^2*(1/52)))))))+exp(3*b[1]*(((2/52)))+b[5]*(((1/52)+3*(1
/52)))).*(((-6)*b[1]^3*b[6]*b[7]^4+3*b[1]^3*b[5]*b[6]*b[7]^4*(1/
52)-3*b[1]^2*b[5]^2*b[6]*b[7]^2*((xt+b[1]*b[6]*(1/52)))+3*b[1]^2
*b[5]^3*b[6]*b[7]^2*(1/52).*((xt+b[1]*b[6]*(1/52)))+b[5]^5*((xt+
```

```
b[1]*b[6]*(1/52))).*((xt^2+2*b[1]*xt*b[6]*(1/52)+(1/52).*((3*b[3
]+b[1]^2*b[6]^2*(1/52))))))))))));

m7=(((1/b[5]^3).*((2*exp((-4)*b[1]*(1/52)-3*b[1]*(1/52)).*(((3*e
xp(2*b[1]*(1/52)-b[5]*(1/52))*b[1]^3*b[6]*b[7]^4*((2+b[5]*(1/52)
+exp(b[5]*(1/52)).*(((-2)+b[5]*(1/52))))))/(2*b[5]^2)+exp(2*b[1]
*(1/52)-b[5]*(1/52)).*((xt+b[1]*b[6]*((exp(b[1]*(1/52)).*(2/52))
))).*((exp(b[5]*(1/52))*b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*((
1+exp(b[5]*(1/52)).*(((-1)+b[5]*(1/52)))))))+1/2*exp((-2)*b[5]*(
((2/52)))).*((xt+b[1]*b[6]*(1/52))).*((exp(2*b[1]*(1/52)+b[5]*((
2*(2/52))))*b[1]^2*b[6]*b[7]^2+exp(2*b[1]*(((2/52)))+b[5]*(((1/5
2)+2*(1/52))))*b[1]^2*b[6]*b[7]^2+exp(2*((b[1]+b[5])).*(((2/52))
)).*((b[5]^3*(1/52).*((b[3]+b[1]^2*(1/52)*b[6]^2))-b[1]^2*b[6]*b
[7]^2+b[1]^2*b[5]*(1/52)*b[6]*b[7]^2))+2*exp(2*b[5]*(((2/52)))+b
[1]*((2*(2/52))))*b[1]*b[5]^3*(1/52)*b[6]*((xt+b[1]*b[6]*(1/52))
)+exp(2*b[1]*(1/52)+2*b[5]*(((2/52)))).*(((-b[1]^2)*b[6]*b[7]^2+
b[1]^2*b[5]*b[6]*b[7]^2*(1/52)+b[5]^3*((xt^2+2*b[1]*xt*b[6]*(1/5
2)+(1/52).*((b[3]+b[1]^2*b[6]^2*(1/52))))))))))))));
 moms=m1~m2~m3~m4~m5~m6~m7;
retp(moms);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_Feed: Returns the obj func for Feed Model
                                *
*****************************************************************
*******************************
* Inputs:       b      -   starting values
                                *
* Output:           -   the objective function to be minimised
                                *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

proc gmm_feed(b);
   local
moms,g1,g2,g3,g4,g5,g6,g7,g8,g9,gsubT,gsubTall,NWlags,NW,invNW,f
lagnw;
        moms=feed_moms(b);
   g1=meanc(xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]);
   g2=meanc(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2]);
   g3=meanc(xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]);
   g4=meanc(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4]);
   g5=meanc(xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5]);
   g6=meanc(xt[2:rows(xt)-2].*(xt[3:rows(xt)-1]^2)
-moms[1:rows(moms)-3,6]);
   g7=meanc((xt[2:rows(xt)-2]^2).*(xt[3:rows(xt)-1])
-moms[1:rows(moms)-3,7]);
   g8=meanc(xt[2:rows(xt)-2].*(xt[4:rows(xt)]^2)
-moms[1:rows(moms)-3,8]);
   g9=meanc((xt[2:rows(xt)-2]^2).*(xt[4:rows(xt)])
-moms[1:rows(moms)-3,9]);
   gsubT=g1|g2|g3|g4|g5|g6|g7|g8|g9;
```

```
   gsubTall=(
              (xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]) ~
(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2])
             ~(xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]) ~
(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4])
             ~(xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5]) ~
(xt[2:rows(xt)-2].*(xt[3:rows(xt)-1]^2) -moms[1:rows(moms)-3,6])
             ~((xt[2:rows(xt)-2]^2).*(xt[3:rows(xt)-1])
-moms[1:rows(moms)-3,7]) ~ (xt[2:rows(xt)-2].*(xt[4:rows(xt)]^2)
-moms[1:rows(moms)-3,8])
             ~((xt[2:rows(xt)-2]^2).*(xt[4:rows(xt)])
-moms[1:rows(moms)-3,9])
             );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;
proc feed_moms(b);
local moms,m1,m2,m3,m4,m5,m6,m7,m8,m9;

m1=(exp((-b[1])*(1/52))*(((((-1)+exp(b[1]*(1/52))))*b[8]*b[3]+b[
1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52))))))/b[1];

m2=(1/(b[1]^2*b[2]^3*b[5]^3))*((exp((-((2*b[1]+b[2]+b[5])))*(1/5
2))*((exp(((2*b[1]+b[2]+b[5]))*(1/52))*b[8]^2*b[2]^3*b[5]^3*b[3]
^2+exp(b[5]*(1/52))*b[1]^2*b[8]^2*b[5]^3*b[3]*b[4]^2+exp(b[2]*(1
/52))*b[1]^4*b[2]^3*b[6]*b[7]^2+2*exp(((b[1]+b[2]+b[5]))*(1/52))
*b[8]*b[2]^3*b[5]^3*b[3]*(((-b[8])*b[3]+b[1]^2*b[6]*(1/52)+b[1]*
((xt-b[8]*b[3]*(1/52)))))+exp(((b[2]+b[5]))*(1/52))*((b[8]^2*b[2
]^3*b[5]^3*b[3]^2+2*b[1]^3*b[2]^3*b[5]^3*b[6]*(1/52)*((xt-b[8]*b
[3]*(1/52)))+2*b[1]*b[8]*b[2]^3*b[5]^3*b[3]*(((-xt)+b[8]*b[3]*(1
/52)))+b[1]^4*b[2]^3*b[6]*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b
[5]*(1/52)))))+b[1]^2*b[5]^3*(((-b[8]^2)*b[3]*b[4]^2+b[8]^2*b[2]
*b[3]*b[4]^2*(1/52)+b[2]^3*((xt^2-2*b[8]*xt*b[3]*(1/52)+b[3]*(1/
52)*((1-2*b[8]*b[6]+b[8]^2*b[3]*(1/52))))))))))))-(2*exp((-((2*
b[1]+b[2])))*(1/52))*b[8]*b[3]*b[9]*b[4]*((1+exp(b[2]*(1/52))*((
(-1)+b[2]*(1/52))))))/b[2]^2;

m3=exp((-3)*b[1]*(1/52))*(((((((-1)+exp(b[1]*(1/52))))*b[8]*b[3]
+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))))^3/b[1]^3+(1/(
2*b[2]^5*b[5]^5))*((3*exp((-((5*b[2]+b[5])))*(1/52))*((exp(b[5]*
(1/52))*b[8]*b[2]*(((-b[2])+b[2]))*b[5]^5*b[3]*b[4]^2-exp(((b[2]
+b[5]))*(1/52))*b[8]*b[2]*(((-b[2])+b[2]))*b[5]^5*b[3]*b[4]^2+2*
exp(5*b[2]*(1/52))*b[1]^3*b[2]^5*b[6]*b[7]^4*((2+b[5]*(1/52)))+e
xp(((5*b[2]+b[5]))*(1/52))*(((-2)*b[8]^3*b[5]^5*b[3]*b[4]^4*(((-
2)+b[2]*(1/52)))+b[8]*b[2]*b[5]^5*b[3]*b[4]^2*((b[2]+b[2]-2*b[2]
^2*(1/52)))+2*b[1]^3*b[2]^5*b[6]*b[7]^4*(((-2)+b[5]*(1/52)))))-e
```

```
xp(((4*b[2]+b[5]))*(1/52))*b[8]*b[5]^5*b[3]*b[4]^2*((b[2]^2+4*b[
8]^2*b[4]^2+b[2]*((b[2]+2*b[8]^2*b[4]^2*(1/52))))))))+(1/(b[1]*
b[2]^3*b[5]^3))*((3*exp((-((b[2]+b[5])))*(1/52))*(((((-1)+exp(b[
1]*(1/52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1
/52)))))*((exp(b[5]*(1/52))*b[8]^2*b[5]^3*b[3]*b[4]^2+exp(b[2]*(
1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2+exp(((b[2]+b[5]))*(1/52))*((b[8
]^2*b[5]^3*b[3]*b[4]^2*(((-1)+b[2]*(1/52)))+b[2]^3*((b[5]^3*b[3]
*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*(1/52))))))))))))+(1/(b[
1]*b[2]^4))*((3*exp((-3)*b[1]*(1/52)-2*b[2]*(1/52))*b[3]*b[9]*b[
4]*(((-2)*exp(((b[1]+b[2]))*(1/52))*b[8]^2*b[2]^2*b[3]-2*exp(((b
[1]+2*b[2]))*(1/52))*b[8]^2*b[2]^2*b[3]*(((-1)+b[2]*(1/52)))+exp
(2*b[2]*(1/52))*((2*b[8]^2*b[2]^2*b[3]*(((-1)+b[2]*(1/52)))-2*b[
1]^2*b[8]*b[2]^2*b[6]*(1/52)*(((-1)+b[2]*(1/52)))+b[1]*(((-6)*b[
8]^2*b[4]^2+b[2]^3*(1/52)*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52)))+b
[8]*b[2]*b[4]*((4*b[9]+3*b[8]*b[4]*(1/52)))-b[2]^2*((1-2*b[8]*xt
+2*b[8]^2*b[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/52)))))))+exp(b[2]*(1/
52))*((2*b[8]^2*b[2]^2*b[3]-2*b[1]^2*b[8]*b[2]^2*b[6]*(1/52)+b[1
]*((6*b[8]^2*b[4]^2+b[8]*b[2]*b[4]*(((-4)*b[9]+3*b[8]*b[4]*(1/52
)))+b[2]^2*((1+2*b[8]^2*b[3]*(1/52)-2*b[8]*((xt+b[9]*b[4]*(1/52)
))))))))))));

m4=exp((-4)*b[1]*(1/52))*(((((((-1)+exp(b[1]*(1/52))))*b[8]*b[3]
+b[1]^2*b[6]*(1/52)

+b[1]*((xt-b[8]*b[3]*(1/52)))))^4/b[1]^4+(1/(b[1]*b[2]^5*b[5]^5)
)*((6*exp((-((5*b[2]+b[5])))*(1/52))

*(((((-1)+exp(b[1]*(1/52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*(
(xt-b[8]*b[3]*(1/52)))))*((exp(b[5]*(1/52))

*b[8]*b[2]*(((-b[2])+b[2]))*b[5]^5*b[3]*b[4]^2-exp(((b[2]+b[5]))
*(1/52))*b[8]*b[2]*(((-b[2])+b[2]))*b[5]^5*b[3]

*b[4]^2+2*exp(5*b[2]*(1/52))*b[1]^3*b[2]^5*b[6]*b[7]^4*((2+b[5]*
(1/52)))+exp(((5*b[2]+b[5]))*(1/52))*(((-2)

*b[8]^3*b[5]^5*b[3]*b[4]^4*(((-2)+b[2]*(1/52)))+b[8]*b[2]*b[5]^5
*b[3]*b[4]^2*((b[2]+b[2]-2*b[2]^2*(1/52)))

+2*b[1]^3*b[2]^5*b[6]*b[7]^4*(((-2)+b[5]*(1/52)))))-exp(((4*b[2]
+b[5]))*(1/52))*b[8]*b[5]^5*b[3]*b[4]^2

*((b[2]^2+4*b[8]^2*b[4]^2+b[2]*((b[2]+2*b[8]^2*b[4]^2*(1/52)))))
))))+(1/(2*b[2]^7*b[5]^7))*((3*exp((-2)

*((b[2]+b[5]))*(1/52))*((exp(2*b[5]*(1/52))*b[8]^4*b[5]^7*b[3]*b
[4]^6+exp(2*b[2]*(1/52))*b[1]^4*b[2]^7*b[6]

*b[7]^6+4*exp(((2*b[2]+b[5]))*(1/52))*b[1]^4*b[2]^7*b[6]*b[7]^6*
((7+5*b[5]*(1/52)+b[5]^2*(1/52)^2))

+exp(2*((b[2]+b[5]))*(1/52))*(((-2)*b[2]^4*b[5]^7*b[3]*b[4]^2-24
*b[8]^2*b[2]^2*b[5]^7*b[3]*b[4]^4

-29*b[8]^4*b[5]^7*b[3]*b[4]^6+2*b[2]^5*b[5]^7*b[3]*b[4]^2*(1/52)
```

```
+12*b[8]^2*b[2]^3*b[5]^7*b[3]*b[4]^4*(1/52)

+10*b[8]^4*b[2]*b[5]^7*b[3]*b[4]^6*(1/52)+b[1]^4*b[2]^7*b[6]*b[7
]^6*(((-29)+10*b[5]*(1/52)))))+2*exp(((b[2]

+2*b[5]))*(1/52))*b[5]^7*b[3]*b[4]^2*((b[2]^4+14*b[8]^4*b[4]^4+6
*b[8]^2*b[2]^3*b[4]^2*(1/52)

+10*b[8]^4*b[2]*b[4]^4*(1/52)+2*b[8]^2*b[2]^2*b[4]^2*((6+b[8]^2*
b[4]^2*(1/52)^2))))))))

+(1/(b[1]^2*b[2]^3*b[5]^3))*((6*exp((-((b[2]+b[5])))*(1/52))*(((
(-1)+exp(b[1]*(1/52))))*b[8]*b[3]

+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))))^2*((exp(b[5]*
(1/52))*b[8]^2*b[5]^3*b[3]*b[4]^2

+exp(b[2]*(1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2+exp(((b[2]+b[5]))*(1
/52))*((b[8]^2*b[5]^3*b[3]*b[4]^2*(((-1)

+b[2]*(1/52)))+b[2]^3*((b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*((
(-1)+b[5]*(1/52)))))))))))

+(1/(b[2]^6*b[5]^6))*((3*exp((-2)*((b[2]+b[5]))*(1/52))*((exp(b[
5]*(1/52))*b[8]^2*b[5]^3*b[3]*b[4]^2

+exp(b[2]*(1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2+exp(((b[2]+b[5]))*(1
/52))*((b[8]^2*b[5]^3*b[3]*b[4]^2*(((-1)

+b[2]*(1/52)))+b[2]^3*((b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*((
(-1)+b[5]*(1/52)))))))))^2))))

-(1/(b[1]^2*b[2]^6*b[5]^3))*((6*exp((-((4*b[1]+2*b[2]+b[5])))*(1
/52))*b[3]*b[9]*b[4]*((2*exp(((2*b[1]

+b[2]+b[5]))*(1/52))*b[8]^3*b[2]^4*b[5]^3*b[3]^2+exp(b[5]*(1/52)
)*b[1]^2*b[8]^2*b[5]^3*b[4]*(((-b[2])*b[9]

+b[8]*b[4]))*((2*b[2]*b[3]+b[4]^2))+2*exp(b[2]*(1/52))*b[1]^4*b[
8]*b[2]^4*b[6]*b[7]^2+2*exp(((2*b[1]+2*b[2]

+b[5]))*(1/52))*b[8]^3*b[2]^4*b[5]^3*b[3]^2*(((-1)+b[2]*(1/52)))
+2*exp(2*b[2]*(1/52))*b[1]^4*b[8]*b[2]^4*b[6]

*b[7]^2*(((-1)+b[2]*(1/52)))+2*exp(((b[1]+2*b[2]+b[5]))*(1/52))*
b[8]*b[2]^2*b[5]^3*b[3]*(((-2)*b[8]^2*b[2]^2

*b[3]*(((-1)+b[2]*(1/52)))+2*b[1]^2*b[8]*b[2]^2*b[6]*(1/52)*(((-
1)+b[2]*(1/52)))+b[1]*((6*b[8]^2*b[4]^2-b[2]^3

*(1/52)*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52)))-b[8]*b[2]*b[4]*((4*
b[9]+3*b[8]*b[4]*(1/52)))+b[2]^2*((1-2*b[8]*xt

+2*b[8]^2*b[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/52)))))))-2*exp(((b[1]
+b[2]+b[5]))*(1/52))*b[8]*b[2]^2*b[5]^3*b[3]
```

```
*((2*b[8]^2*b[2]^2*b[3]-2*b[1]^2*b[8]*b[2]^2*b[6]*(1/52)+b[1]*((
6*b[8]^2*b[4]^2+b[8]*b[2]*b[4]*(((-4)*b[9]

+3*b[8]*b[4]*(1/52)))+b[2]^2*((1+2*b[8]^2*b[3]*(1/52)-2*b[8]*((x
t+b[9]*b[4]*(1/52)))))))))+exp(((2*b[2]+b[5]))

*(1/52))*((2*b[8]^3*b[2]^4*b[5]^3*b[3]^2*(((-1)+b[2]*(1/52)))+2*
b[1]^4*b[8]*b[2]^4*b[6]*(((-1)+b[2]*(1/52)))

*((b[5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))))+2*b[1]*b[8
]*b[2]^2*b[5]^3*b[3]*(((-6)*b[8]^2*b[4]^2

+b[2]^3*(1/52)*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52)))+b[8]*b[2]*b[
4]*((4*b[9]+3*b[8]*b[4]*(1/52)))-b[2]^2

*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/52)))))-
2*b[1]^3*b[2]^2*b[5]^3*b[6]*(1/52)*(((-6)

*b[8]^2*b[4]^2+b[2]^3*(1/52)*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52))
)+b[8]*b[2]*b[4]*((4*b[9]+3*b[8]*b[4]*(1/52)))

-b[2]^2*((1-2*b[8]*xt+2*b[8]^2*b[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/5
2)))))+b[1]^2*b[5]^3*(((-29)*b[8]^3*b[4]^4

+2*b[2]^5*(1/52)*((b[8]*xt^2+b[8]*b[3]*(1/52)*((2-2*b[8]*b[6]+b[
8]^2*b[3]*(1/52)))-xt*((1+2*b[8]^2*b[3]

*(1/52)))))+2*b[2]^3*b[4]*((2*b[8]*b[9]^2*b[4]*(1/52)+b[8]*b[4]*
(1/52)*((3-3*b[8]*xt+4*b[8]^2*b[3]*(1/52)))

+b[9]*((2-4*b[8]*xt+6*b[8]^2*b[3]*(1/52)))))+b[8]^2*b[2]*b[4]^2*
((35*b[9]*b[4]+2*b[8]*((b[3]+5*b[4]^2

*(1/52)))))-2*b[8]*b[2]^2*b[4]*((6*((1+b[9]^2))*b[4]+8*b[8]^2*b[
3]*b[4]*(1/52)+b[8]*((b[3]*b[9]-6*xt*b[4]

+6*b[9]*b[4]^2*(1/52)))))-2*b[2]^4*((b[8]*xt^2-xt*((1+2*b[8]^2*b
[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/52)))

+(1/52)*((2*b[8]*b[3]+b[9]*b[4]+b[8]^3*b[3]^2*(1/52)+b[8]^2*b[3]
*(((-2)*b[6]+3*b[9]*b[4]*(1/52)))))))))))

+exp(((b[2]+b[5]))*(1/52))*((2*b[8]^3*b[2]^4*b[5]^3*b[3]^2+2*b[1
]^4*b[8]*b[2]^4*b[6]*((b[5]^3*b[6]*(1/52)^2

+b[7]^2*(((-1)+b[5]*(1/52)))))+2*b[1]*b[8]*b[2]^2*b[5]^3*b[3]*((
6*b[8]^2*b[4]^2+b[8]*b[2]*b[4]*(((-4)*b[9]

+3*b[8]*b[4]*(1/52)))+b[2]^2*((1+2*b[8]^2*b[3]*(1/52)-2*b[8]*((x
t+b[9]*b[4]*(1/52)))))))-2*b[1]^3*b[2]^2

*b[5]^3*b[6]*(1/52)*((6*b[8]^2*b[4]^2+b[8]*b[2]*b[4]*(((-4)*b[9]
+3*b[8]*b[4]*(1/52)))+b[2]^2*((1+2*b[8]^2
```

```
*b[3]*(1/52)-2*b[8]*((xt+b[9]*b[4]*(1/52)))))))+b[1]^2*b[5]^3*((
28*b[8]^3*b[4]^4+2*b[8]^2*b[2]*b[4]^2

*(((-17)*b[9]*b[4]-2*b[8]*((b[3]-5*b[4]^2*(1/52)))))+4*b[8]*b[2]
^2*b[4]*((3*((1+b[9]^2))*b[4]+b[8]^2*b[4]

*(1/52)*((4*b[3]+b[4]^2*(1/52)))+b[8]*((b[3]*b[9]-3*b[4]*((xt+2*
b[9]*b[4]*(1/52)))))))+b[2]^3*b[4]*((8*b[8]

*b[9]^2*b[4]*(1/52)+6*b[8]*b[4]*(1/52)*((1-b[8]*xt+b[8]^2*b[3]*(
1/52)))-b[9]*((4-8*b[8]*xt+b[8]^2*(1/52)

*((12*b[3]+5*b[4]^2*(1/52)))))+2*b[2]^4*((b[8]*xt^2+xt*(((-1)-
2*b[8]^2*b[3]*(1/52)+2*b[8]*b[9]*b[4]*(1/52)))

+(1/52)*(((-b[9])*b[4]+b[8]^3*b[3]^2*(1/52)-2*b[8]^2*b[3]*((b[6]
+b[9]*b[4]*(1/52)))+b[8]*((2*b[3]+b[9]^2*b[4]^2
    *(1/52)))))))))))))));

m5=(((1/(2*b[1]^2*b[2]^3*b[5]^3)).*((exp((-2).*((b[2]+b[5])).*(1
/52)-b[1]*(((1/52)+2*(1/52)))).*((2*exp(2*((b[2]+b[5])).*(1/52)+
b[1]*(((1/52)+2*(1/52))))*b[8]^2*b[2]^3*b[5]^3*b[3]^2+2*exp(2*((
b[1]+b[2]+b[5])).*(1/52))*b[1]*b[8]*b[2]^3*b[5]^3*(1/52)*b[3]*((
(-b[8])*b[3]+b[1]*b[6]))+exp(2*b[5]*(1/52))*b[1]^2*b[8]^2*b[5]^3
*b[3]*((b[4]^2-b[4]^2))-2*exp(((b[2]+2*b[5])).*(1/52))*b[1]^2*b[
8]*b[5]^3*b[3]*((2*b[2]*b[9]*b[4]+b[8]*((b[4]^2-2*b[4]^2))))+2*e
xp(((2*b[2]+b[5])).*(1/52))*b[1]^4*b[2]^3*b[6]*b[7]^2+2*exp(2*((
b[2]+b[5])).*(1/52)+b[1]*(((1/52)+(1/52))))*b[8]*b[2]^3*b[5]^3*b
[3]*(((-b[8])*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)
))))+2*exp(((b[1]+2*((b[2]+b[5])))).*(1/52))*b[2]^3*b[5]^3*((b[8
]*((b[3]-b[1]*(1/52)*b[3]))+b[1]^2*(1/52)*b[6])).*(((-b[8])*b[3]
+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))))+exp(2*((b[2]+
b[5])).*(1/52)).*((2*b[8]^2*b[2]^3*b[5]^3*b[3]^2+4*b[1]^3*b[2]^3
*b[5]^3*b[6]*(1/52).*((xt-b[8]*b[3]*(1/52)))+4*b[1]*b[8]*b[2]^3*
b[5]^3*b[3]*(((-xt)+b[8]*b[3]*(1/52)))+2*b[1]^4*b[2]^3*b[6]*((b[
5]^3*b[6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))))+b[1]^2*b[5]^3*(
(b[8]^2*b[3]*((b[4]^2-3*b[4]^2))-4*b[8]*b[2]^2*b[3]*b[9]*b[4]*(1
/52)+2*b[8]*b[2]*b[3]*b[4]*((2*b[9]+b[8]*b[4]*(1/52)))+2*b[2]^3*
((xt^2-2*b[8]*xt*b[3]*(1/52)+b[3]*(1/52).*((1-2*b[8]*b[6]+b[8]^2
*b[3]*(1/52)))))))))))))));

m6=(1/4*((((1/(b[1]*b[2]^3*b[5]^3)).*((4*exp((-2).*((b[2]+b[5]))
.*(1/52)-b[1]*((2*(1/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(((1/52)
+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*((exp(b[1]*(1/52)).*(1/52)+(1
/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1]*(1/52)).*(1/52)+(1/52)))))
)).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b[3]*((b[4]^2-b[4]^2))-2*
exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3*b[3]*((2*b[2]*b[9]*b[4]
+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[2]+b[5])).*(1/52))*b[1]^
2*b[2]^3*b[6]*b[7]^2+exp(2*((b[2]+b[5])).*(1/52)).*(((-4)*b[8]*b
[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))+b[8]^2*b[5]^3*b[3
]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))))+2*b[2]^3*((b[5]^3*b[3
]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*(1/52)))))))))))+((1/(b
[2]^5*b[5]^5))*((2*exp((-3).*((b[2]+b[5])).*(1/52)-b[1]*((2*(1/5
2)+3*(1/52)))).*((exp(3*b[5]*(1/52))*b[8]^3*b[5]^5*b[3]*((b[4]^4
-3*b[4]^2*b[4]^2+2*b[4]^4))+6*exp(3*b[2]*(1/52)+2*b[5]*(1/52))*b
```

```
[1]^3*b[2]^5*b[6]*b[7]^4*((2+b[5]*(1/52)))-3*exp(((b[2]+3*b[5]))
.*(1/52))*b[8]*b[5]^5*b[3]*((b[4]^2-b[4]^2)).*((b[8]^2*b[4]^2+2*
b[8]*b[2]*b[4]*(((-b[9])+b[8]*b[4]*(1/52)))+b[2]^2*((1-2*b[8]*b[
9]*b[4]*(1/52)))))+exp(3*((b[2]+b[5])).*(1/52)).*((6*b[2]^3*(((-
2)*b[1]^3*b[2]^2*b[6]*b[7]^4+b[1]^3*b[2]^2*b[5]*b[6]*b[7]^4*(1/5
2)+b[5]^5*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))))-b[8]^3*b[5]^5*b[
3]*((b[4]^4+3*b[4]^2*b[4]^2+2*b[4]^4*(((-8)+3*b[2]*(1/52)))))+6*
b[8]^2*b[2]*b[5]^5*b[3]*b[9]*b[4]*((b[4]^2+b[4]^2*(((-7)+3*b[2]*
(1/52)))))-3*b[8]*b[2]^2*b[5]^5*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[
2]*(1/52)+4*b[9]^2*(((-2)+b[2]*(1/52)))))))))+3*exp(2*b[2]*(1/52
)+3*b[5]*(1/52))*b[5]^5*b[3]*((2*b[2]^3*b[9]*b[4]-2*b[8]^2*b[2]*
b[9]*b[4]*((b[4]^2-4*b[4]^2)).*((2+b[2]*(1/52)))+b[8]^3*((b[4]^2
-2*b[4]^2)).*((b[4]^2+b[4]^2*((3+2*b[2]*(1/52)))))-2*b[8]*b[2]^2
*(((-b[4]^2)+2*b[4]^2*((1+b[9]^2*((2+b[2]*(1/52)))))))))))))))-((
1/b[1])*((2*exp((-b[1]).*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[
8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52))))).*(((-(
(2*b[8]^2*b[3]^2)/b[1]^2))+(4*exp((-b[1]).*(1/52))*b[8]*(1/52)*b
[3]*((b[8]*b[3]-b[1]*b[6])))/b[1]-(exp((-2).*((b[1]+b[2])).*(1/5
2))*b[8]^2*b[3]*((b[4]^2-b[4]^2)))/b[2]^3-(exp((-2).*((b[2]*(1/5
2)+b[1]*(((1/52)+(1/52))))))*b[8]^2*b[3]*((b[4]^2-b[4]^2)))/b[2]
^3+(2*exp((-((2*b[1]+b[2])))).*(1/52))*b[8]*b[3]*((2*b[2]*b[9]*b[
4]+b[8]*((b[4]^2-2*b[4]^2)))))/b[2]^3+(2*exp((-b[2]).*(1/52)-2*b
[1]*(((1/52)+(1/52))))*b[8]*b[3]*((2*b[2]*b[9]*b[4]+b[8]*((b[4]^
2-2*b[4]^2)))))/b[2]^3-(2*exp((-((2*b[1]+b[5])))).*(1/52))*b[1]^2
*b[6]*b[7]^2)/b[5]^3-(2*exp((-b[5]).*(1/52)-2*b[1]*(((1/52)+(1/5
2))))*b[1]^2*b[6]*b[7]^2)/b[5]^3+((1/(b[2]^3*b[5]^3))*((exp((-2)
*b[1]*(1/52)).*((4*b[8]*b[2]^2*b[5]^3*(1/52)*b[3]*b[9]*b[4]-2*b[
8]*b[2]*b[5]^3*b[3]*b[4]*((2*b[9]+b[8]*(1/52)*b[4]))-b[8]^2*b[5]
^3*b[3]*((b[4]^2-3*b[4]^2))-2*b[2]^3*((b[5]^3*(1/52).*((b[3]+b[8
]^2*(1/52)*b[3]^2-2*b[1]*b[8]*(1/52)*b[3]*b[6]+b[1]^2*(1/52)*b[6
]^2))-b[1]^2*b[6]*b[7]^2+b[1]^2*b[5]*(1/52)*b[6]*b[7]^2)))))))+(
4*exp((-b[1]).*(((1/52)+(1/52))))*b[8]*b[3]*(((-b[1])*xt+b[8]*b[
3]+b[1]*b[8]*b[3]*(1/52)-b[1]^2*b[6]*(1/52))))/b[1]^2-(4*exp((-b
[1]).*((2*(1/52)+(1/52)))).*(1/52).*(((-b[8])*b[3]+b[1]*b[6])).*
(((-b[8])*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))))
)/b[1]+((1/(b[1]^2*b[2]^3*b[5]^3))*((exp((-2)*b[1]*(((1/52)+(1/5
2)))).*(((-2)*b[8]^2*b[2]^3*b[5]^3*b[3]^2+4*b[1]*b[8]*b[2]^3*b[5
]^3*b[3]*((xt-b[8]*b[3]*(1/52)))+4*b[1]^3*b[2]^3*b[5]^3*b[6]*(1/
52).*(((-xt)+b[8]*b[3]*(1/52)))-2*b[1]^4*b[2]^3*b[6]*((b[5]^3*b[
6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))))-b[1]^2*b[5]^3*((b[8]^2
*b[3]*((b[4]^2-3*b[4]^2))-4*b[8]*b[2]^2*b[3]*b[9]*b[4]*(1/52)+2*
b[8]*b[2]*b[3]*b[4]*((2*b[9]+b[8]*b[4]*(1/52)))+2*b[2]^3*((xt^2-
2*b[8]*xt*b[3]*(1/52)+b[3]*(1/52).*((1-2*b[8]*b[6]+b[8]^2*b[3]*(
1/52))))))))))))))))))));

m7=(1/2*(((2*exp((-b[1]).*(((1/52)+3*(1/52)))).*(((((-1)+exp(b[1
]*(1/52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/
52)))))^2
.*(((((-1)+exp(b[1]*(((1/52)+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*(
(exp(b[1]*(1/52)).*(1/52)+(1/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1
]*(1/52)).*(1/52)+(1/52)))))))/b[1]^3+((1/(b[1]*b[2]^3*b[5]^3))
*((exp((-2).*((b[2]+b[5])).*(1/52)-b[1]*(((1/52)+3*(1/52)))).*((
(((-1)+exp(b[1]*(((1/52)+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*((exp
(b[1]*(1/52)).*(1/52)+(1/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1]*(1
/52)).*(1/52)+(1/52)))))))).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b
```

```
[3]*((b[4]^2-b[4]^2))-2*exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3
*b[3]*((2*b[2]*b[9]*b[4]+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[
2]+b[5])).*(1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2+exp(2*((b[2]+b[5]))
.*(1/52)).*(((-4)*b[8]*b[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1
/52)))+b[8]^2*b[5]^3*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))
))+2*b[2]^3*((b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*
(1/52)))))))))))+((1/(b[1]*b[2]^3*b[5]^3))*((2*exp((-2).*((b[2]
+b[5])).*(1/52)-b[1]*(((1/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(1/
52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))
)).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b[3]*((b[4]^2-b[4]^2))-2*
exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3*b[3]*((2*b[2]*b[9]*b[4]
+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[2]+b[5])).*(1/52))*b[1]^
2*((b[2]^2))^(3/2)*b[6]*b[7]^2+exp(2*((b[2]+b[5])).*(1/52)).*(((
-4)*b[8]*b[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))+b[8]^2*
b[5]^3*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))))+2*b[2]^3*((
b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*(1/52)))))))))
)))+((1/(b[2]^5*b[5]^5))*((exp((-3).*((b[2]+b[5])).*(1/52)-b[1]*
((2*(1/52)+7*(1/52)))).*((exp(3*b[5]*(1/52)+b[1]*(((1/52)+4*(1/5
2))))*b[8]^3*b[5]^5*b[3]*((b[4]^4-3*b[4]^2*b[4]^2+2*b[4]^4))+6*e
xp(b[1]*(1/52)+4*b[1]*(1/52)+3*b[2]*(1/52)+2*b[5]*(1/52))*b[1]^3
*b[2]^5*b[6]*b[7]^4*((2+b[5]*(1/52)))-3*exp(b[1]*(1/52)+4*b[1]*(
1/52)+b[2]*(1/52)+3*b[5]*(1/52))*b[8]*b[5]^5*b[3]*((b[4]^2-b[4]^
2)).*((b[8]^2*b[4]^2+2*b[8]*b[2]*b[4]*(((-b[9])+b[8]*b[4]*(1/52)
))+b[2]^2*((1-2*b[8]*b[9]*b[4]*(1/52)))))+exp(3*((b[2]+b[5])).*(
1/52)+b[1]*(((1/52)+4*(1/52)))).*((6*b[2]^3*(((-2)*b[1]^3*b[2]^2
*b[6]*b[7]^4+b[1]^3*b[2]^2*b[5]*b[6]*b[7]^4*(1/52)+b[5]^5*b[3]*b
[9]*b[4]*(((-1)+b[2]*(1/52))))))-b[8]^3*b[5]^5*b[3]*((b[4]^4+3*b[
4]^2*b[4]^2+2*b[4]^4*(((-8)+3*b[2]*(1/52)))))+6*b[8]^2*b[2]*b[5]
^5*b[3]*b[9]*b[4]*((b[4]^2+b[4]^2*(((-7)+3*b[2]*(1/52)))))-3*b[8
]*b[2]^2*b[5]^5*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)+4*b[9]
^2*(((-2)+b[2]*(1/52)))))))))+3*exp(b[1]*(1/52)+4*b[1]*(1/52)+2*
b[2]*(1/52)+3*b[5]*(1/52))*b[5]^5*b[3]*((2*b[2]^3*b[9]*b[4]-2*b[
8]^2*b[2]*b[9]*b[4]*((b[4]^2-4*b[4]^2)).*((2+b[2]*(1/52)))+b[8]^
3*((b[4]^2-2*b[4]^2)).*((b[4]^2+b[4]^2*((3+2*b[2]*(1/52)))))-2*b
[8]*b[2]^2*(((-b[4]^2)+2*b[4]^2*((1+b[9]^2*((2+b[2]*(1/52)))))))
))))))))));

m8=(1/4*((((1/(b[1]*b[2]^3*b[5]^3)).*((4*exp((-2).*((b[2]+b[5]))
.*(1/52)-b[1]*((2*(2/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(((2/52)
+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*((exp(b[1]*(1/52)).*(2/52)+(1
/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1]*(1/52)).*(2/52)+(1/52)))))
)).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b[3]*((b[4]^2-b[4]^2))-2*
exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3*b[3]*((2*b[2]*b[9]*b[4]
+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[2]+b[5])).*(1/52))*b[1]^
2*b[2]^3*b[6]*b[7]^2+exp(2*((b[2]+b[5])).*(1/52)).*(((-4)*b[8]*b
[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))+b[8]^2*b[5]^3*b[3
]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))))+2*b[2]^3*((b[5]^3*b[3
]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*(1/52)))))))))))+((1/(b
[2]^5*b[5]^5))*((2*exp((-3).*((b[2]+b[5])).*(1/52)-b[1]*((2*(2/5
2)+3*(1/52)))).*((exp(3*b[5]*(1/52))*b[8]^3*b[5]^5*b[3]*((b[4]^4
-3*b[4]^2*b[4]^2+2*b[4]^4))+6*exp(3*b[2]*(1/52)+2*b[5]*(1/52))*b
[1]^3*b[2]^5*b[6]*b[7]^4*((2+b[5]*(1/52)))-3*exp(((b[2]+3*b[5]))
.*(1/52))*b[8]*b[5]^5*b[3]*((b[4]^2-b[4]^2)).*((b[8]^2*b[4]^2+2*
b[8]*b[2]*b[4]*(((-b[9])+b[8]*b[4]*(1/52)))+b[2]^2*((1-2*b[8]*b[
9]*b[4]*(1/52)))))+exp(3*((b[2]+b[5])).*(1/52)).*((6*b[2]^3*(((-
```

```
2)*b[1]^3*b[2]^2*b[6]*b[7]^4+b[1]^3*b[2]^2*b[5]*b[6]*b[7]^4*(1/5
2)+b[5]^5*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))))-b[8]^3*b[5]^5*b[
3]*((b[4]^4+3*b[4]^2*b[4]^2+2*b[4]^4*(((-8)+3*b[2]*(1/52)))))+6*
b[8]^2*b[2]*b[5]^5*b[3]*b[9]*b[4]*((b[4]^2+b[4]^2*(((-7)+3*b[2]*
(1/52)))))-3*b[8]*b[2]^2*b[5]^5*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[
2]*(1/52)+4*b[9]^2*(((-2)+b[2]*(1/52)))))))))+3*exp(2*b[2]*(1/52
)+3*b[5]*(1/52))*b[5]^5*b[3]*((2*b[2]^3*b[9]*b[4]-2*b[8]^2*b[2]*
b[9]*b[4]*((b[4]^2-4*b[4]^2)).*((2+b[2]*(1/52)))+b[8]^3*((b[4]^2
-2*b[4]^2)).*((b[4]^2+b[4]^2*((3+2*b[2]*(1/52)))))-2*b[8]*b[2]^2
*(((-b[4]^2)+2*b[4]^2*((1+b[9]^2*((2+b[2]*(1/52)))))))))))))))-((
1/b[1])*((2*exp((-b[1]).*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[
8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52))))).*(((-(
(2*b[8]^2*b[3]^2)/b[1]^2))+(4*exp((-b[1]).*(2/52))*b[8]*(2/52)*b
[3]*((b[8]*b[3]-b[1]*b[6])))/b[1]-(exp((-2).*((b[1]+b[2])).*(2/5
2))*b[8]^2*b[3]*((b[4]^2-b[4]^2)))/b[2]^3-(exp((-2).*((b[2]*(1/5
2)+b[1]*(((2/52)+(1/52))))))*b[8]^2*b[3]*((b[4]^2-b[4]^2)))/b[2]
^3+(2*exp((-((2*b[1]+b[2])))).*(2/52))*b[8]*b[3]*((2*b[2]*b[9]*b[
4]+b[8]*((b[4]^2-2*b[4]^2)))))/b[2]^3+(2*exp((-b[2]).*(1/52)-2*b
[1]*(((2/52)+(1/52))))*b[8]*b[3]*((2*b[2]*b[9]*b[4]+b[8]*((b[4]^
2-2*b[4]^2)))))/b[2]^3-(2*exp((-((2*b[1]+b[5])))).*(2/52))*b[1]^2
*b[6]*b[7]^2)/b[5]^3-(2*exp((-b[5]).*(1/52)-2*b[1]*(((2/52)+(1/5
2))))*b[1]^2*b[6]*b[7]^2)/b[5]^3+((1/(b[2]^3*b[5]^3))*((exp((-2)
*b[1]*(2/52)).*((4*b[8]*b[2]^2*b[5]^3*(2/52)*b[3]*b[9]*b[4]-2*b[
8]*b[2]*b[5]^3*b[3]*b[4]*((2*b[9]+b[8]*(2/52)*b[4]))-b[8]^2*b[5]
^3*b[3]*((b[4]^2-3*b[4]^2))-2*b[2]^3*((b[5]^3*(2/52).*((b[3]+b[8
]^2*(2/52)*b[3]^2-2*b[1]*b[8]*(2/52)*b[3]*b[6]+b[1]^2*(2/52)*b[6
]^2))-b[1]^2*b[6]*b[7]^2+b[1]^2*b[5]*(2/52)*b[6]*b[7]^2)))))))+(
4*exp((-b[1]).*(((2/52)+(1/52))))*b[8]*b[3]*(((-b[1])*xt+b[8]*b[
3]+b[1]*b[8]*b[3]*(1/52)-b[1]^2*b[6]*(1/52))))/b[1]^2-(4*exp((-b
[1]).*((2*(2/52)+(1/52)))).*(2/52).*(((-b[8])*b[3]+b[1]*b[6])).*
(((-b[8])*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))))
)/b[1]+((1/(b[1]^2*b[2]^3*b[5]^3))*((exp((-2)*b[1]*(((2/52)+(1/5
2)))).*(((-2)*b[8]^2*b[2]^3*b[5]^3*b[3]^2+4*b[1]*b[8]*b[2]^3*b[5
]^3*b[3]*((xt-b[8]*b[3]*(1/52)))+4*b[1]^3*b[2]^3*b[5]^3*b[6]*(1/
52).*(((-xt)+b[8]*b[3]*(1/52)))-2*b[1]^4*b[2]^3*b[6]*((b[5]^3*b[
6]*(1/52)^2+b[7]^2*(((-1)+b[5]*(1/52)))))-b[1]^2*b[5]^3*((b[8]^2
*b[3]*((b[4]^2-3*b[4]^2))-4*b[8]*b[2]^2*b[3]*b[9]*b[4]*(1/52)+2*
b[8]*b[2]*b[3]*b[4]*((2*b[9]+b[8]*b[4]*(1/52)))+2*b[2]^3*((xt^2-
2*b[8]*xt*b[3]*(1/52)+b[3]*(1/52).*((1-2*b[8]*b[6]+b[8]^2*b[3]*(
1/52))))))))))))))))));

m9=(1/2*(((2*exp((-b[1]).*(((2/52)+3*(1/52)))).*(((((-1)+exp(b[1
]*(1/52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/
52)))))^2
.*(((((-1)+exp(b[1]*(((2/52)+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*(
(exp(b[1]*(1/52)).*(2/52)+(1/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1
]*(1/52)).*(2/52)+(1/52)))))))/b[1]^3+((1/(b[1]*b[2]^3*b[5]^3))
*((exp((-2).*((b[2]+b[5])).*(1/52)-b[1]*(((2/52)+3*(1/52)))).*((
(((-1)+exp(b[1]*(((2/52)+(1/52))))))*b[8]*b[3]+b[1]^2*b[6]*((exp
(b[1]*(1/52)).*(2/52)+(1/52)))+b[1]*((xt-b[8]*b[3]*((exp(b[1]*(1
/52)).*(2/52)+(1/52)))))).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b
[3]*((b[4]^2-b[4]^2))-2*exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3
*b[3]*((2*b[2]*b[9]*b[4]+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[
2]+b[5])).*(1/52))*b[1]^2*b[2]^3*b[6]*b[7]^2+exp(2*((b[2]+b[5]))
.*(1/52)).*(((-4)*b[8]*b[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1
```

```
/52)))+b[8]^2*b[5]^3*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))
))+2*b[2]^3*((b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*
(1/52)))))))))))+((1/(b[1]*b[2]^3*b[5]^3))*((2*exp((-2).*((b[2]
+b[5])).*(1/52)-b[1]*(((2/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(1/
52))))*b[8]*b[3]+b[1]^2*b[6]*(1/52)+b[1]*((xt-b[8]*b[3]*(1/52)))
)).*((exp(2*b[5]*(1/52))*b[8]^2*b[5]^3*b[3]*((b[4]^2-b[4]^2))-2*
exp(((b[2]+2*b[5])).*(1/52))*b[8]*b[5]^3*b[3]*((2*b[2]*b[9]*b[4]
+b[8]*((b[4]^2-2*b[4]^2))))+2*exp(((2*b[2]+b[5])).*(1/52))*b[1]^
2*((b[2]^2))^(3/2)*b[6]*b[7]^2+exp(2*((b[2]+b[5])).*(1/52)).*(((
-4)*b[8]*b[2]*b[5]^3*b[3]*b[9]*b[4]*(((-1)+b[2]*(1/52)))+b[8]^2*
b[5]^3*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)))))+2*b[2]^3*((
b[5]^3*b[3]*(1/52)+b[1]^2*b[6]*b[7]^2*(((-1)+b[5]*(1/52)))))))))
)))+((1/(b[2]^5*b[5]^5))*((exp((-3).*((b[2]+b[5])).*(1/52)-b[1]*
((2*(2/52)+7*(1/52)))).*((exp(3*b[5]*(1/52)+b[1]*(((2/52)+4*(1/5
2))))*b[8]^3*b[5]^5*b[3]*((b[4]^4-3*b[4]^2*b[4]^2+2*b[4]^4))+6*e
xp(b[1]*(2/52)+4*b[1]*(1/52)+3*b[2]*(1/52)+2*b[5]*(1/52))*b[1]^3
*b[2]^5*b[6]*b[7]^4*((2+b[5]*(1/52)))-3*exp(b[1]*(2/52)+4*b[1]*(
1/52)+b[2]*(1/52)+3*b[5]*(1/52))*b[8]*b[5]^5*b[3]*((b[4]^2-b[4]^
2)).*((b[8]^2*b[4]^2+2*b[8]*b[2]*b[4]*(((-b[9])+b[8]*b[4]*(1/52)
))+b[2]^2*((1-2*b[8]*b[9]*b[4]*(1/52)))))+exp(3*((b[2]+b[5])).*(
1/52)+b[1]*(((2/52)+4*(1/52)))).*((6*b[2]^3*(((-2)*b[1]^3*b[2]^2
*b[6]*b[7]^4+b[1]^3*b[2]^2*b[5]*b[6]*b[7]^4*(1/52)+b[5]^5*b[3]*b
[9]*b[4]*(((-1)+b[2]*(1/52)))))-b[8]^3*b[5]^5*b[3]*((b[4]^4+3*b[
4]^2*b[4]^2+2*b[4]^4*(((-8)+3*b[2]*(1/52)))))+6*b[8]^2*b[2]*b[5]
^5*b[3]*b[9]*b[4]*((b[4]^2+b[4]^2*(((-7)+3*b[2]*(1/52)))))-3*b[8
]*b[2]^2*b[5]^5*b[3]*((b[4]^2+b[4]^2*(((-3)+2*b[2]*(1/52)+4*b[9]
^2*(((-2)+b[2]*(1/52)))))))))+3*exp(b[1]*(2/52)+4*b[1]*(1/52)+2*
b[2]*(1/52)+3*b[5]*(1/52))*b[5]^5*b[3]*((2*b[2]^3*b[9]*b[4]-2*b[
8]^2*b[2]*b[9]*b[4]*((b[4]^2-4*b[4]^2)).*((2+b[2]*(1/52)))+b[8]^
3*((b[4]^2-2*b[4]^2)).*((b[4]^2+b[4]^2*((3+2*b[2]*(1/52)))))-2*b
[8]*b[2]^2*(((-b[4]^2)+2*b[4]^2*((1+b[9]^2*((2+b[2]*(1/52)))))))
))))))))));
 moms=m1~m2~m3~m4~m5~m6~m7~m8~m9;
retp(moms);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_SVJ: Returns the obj func for SV Model
                                  *
*****************************************************************
*******************************
* Inputs:       b      -   starting values
                                 *
* Output:           -   the objective function to be minimised
                                   *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_SVJ(b);
   local
moms,g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,gsubT,gsubTall,NWlags,NW,inv
NW,flagnw,g11,g12,g13,g14;
        moms=SVJ_moms(b);
   g1=meanc(xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]);
   g2=meanc(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2]);
```

```
   g3=meanc(xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]);
   g4=meanc(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4]);
   g5=meanc(xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5]);
   g6=meanc(xt[2:rows(xt)-2].*xt[4:rows(xt)]
-moms[1:rows(moms)-3,6]);
   g7=meanc(xt[2:rows(xt)-2].*(xt[3:rows(xt)-1]^2)
-moms[1:rows(moms)-3,7]);
   g8=meanc((xt[2:rows(xt)-2]^2).*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,8]);
   g9=meanc(xt[2:rows(xt)-2].*(xt[4:rows(xt)]^2)
-moms[1:rows(moms)-3,9]);
   g10=meanc((xt[2:rows(xt)-2]^2).*xt[4:rows(xt)]
-moms[1:rows(moms)-3,10]);
   gsubT=g1|g2|g3|g4|g5|g6|g7|g8|g9|g10;
   gsubTall=((xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]) ~
(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2])
            ~(xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]) ~
(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4])
            ~(xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5]) ~ (xt[2:rows(xt)-2].*xt[4:rows(xt)]
-moms[1:rows(moms)-3,6])
            ~(xt[2:rows(xt)-2].*(xt[3:rows(xt)-1]^2)
-moms[1:rows(moms)-3,7])
~((xt[2:rows(xt)-2]^2).*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,8])
            ~(xt[2:rows(xt)-2].*(xt[4:rows(xt)]^2)
-moms[1:rows(moms)-3,9])~((xt[2:rows(xt)-2]^2).*xt[4:rows(xt)]
-moms[1:rows(moms)-3,10])           );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;
proc SVJ_moms(b);
local
moms,m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,phi,c,d,Kapa
r,R_bar,Kapa_v,v_bar,Sigma_v,Rho,iota,xi,r1,r2,Lamdau,NUu,Lamdad
,NUd,u;
    m1=exp((-b[1]).*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[2]+
xt))-(((((-1)+exp(b[1]*(1/52))))).*((b[10]*b[9]-b[8]*b[7])))))/(b[
1]*exp(b[1]*(1/52)));

m2=(1/b[1]^2).*((exp((-2)*b[1]*(1/52)).*(((((-1)+exp(b[1]*(1/52)
)))^2*b[2]^2*b[1]^2+xt^2*b[1]^2-b[10]^2*b[1]*b[9]+exp(2*b[1]*(1/
52))*b[10]^2*b[1]*b[9]+b[10]^2*b[9]^2-2*exp(b[1]*(1/52))*b[10]^2
*b[9]^2+exp(2*b[1]*(1/52))*b[10]^2*b[9]^2-b[8]^2*b[1]*b[7]+exp(2
*b[1]*(1/52))*b[8]^2*b[1]*b[7]-2*b[10]*b[8]*b[9]*b[7]+4*exp(b[1]
*(1/52))*b[10]*b[8]*b[9]*b[7]-2*exp(2*b[1]*(1/52))*b[10]*b[8]*b[
```

```
9]*b[7]+b[8]^2*b[7]^2-2*exp(b[1]*(1/52))*b[8]^2*b[7]^2+exp(2*b[1
]*(1/52))*b[8]^2*b[7]^2-2*(((-1)+exp(b[1]*(1/52))))*xt*b[1]*((b[
10]*b[9]-b[8]*b[7]))+2*(((-1)+exp(b[1]*(1/52))))*b[2]*b[1]*((xt*
b[1]-(((-1)+exp(b[1]*(1/52))))).*((b[10]*b[9]-b[8]*b[7]))))+b[4]*
b[1]^2*(1/52)))));

m3=(1/(2*b[1]^3*b[3]^2)).*((exp((-((3*b[1]+3*b[3]))).*(1/52)).*(
((-6)*exp(2*b[1]*(1/52)+3*b[3]*(1/52))*b[3]^2*((b[2]*b[1]-xt*b[1
]-b[10]*b[9]+b[8]*b[7])).*((b[2]^2*b[1]^2+b[10]^2*b[9]*((b[1]+b[
9]))-2*b[10]*b[8]*b[9]*b[7]+b[8]^2*b[7]*((b[1]+b[7]))+b[2]*(((-2
)*b[10]*b[1]*b[9]+2*b[8]*b[1]*b[7])))))+2*exp(3*((b[1]+b[3])).*(1
/52))*b[3]^2*((b[2]^3*b[1]^3-b[10]^3*b[9]*((2*b[1]^2+3*b[1]*b[9]
+b[9]^2))+3*b[10]^2*b[8]*b[9]*((b[1]+b[9]))*b[7]-3*b[10]*b[8]^2*
b[9]*b[7]*((b[1]+b[7]))+3*b[2]^2*b[1]^2*(((-b[10])*b[9]+b[8]*b[7
]))+b[8]^3*b[7]*((2*b[1]^2+3*b[1]*b[7]+b[7]^2))+3*b[2]*b[1]*((b[
10]^2*b[9]*((b[1]+b[9]))-2*b[10]*b[8]*b[9]*b[7]+b[8]^2*b[7]*((b[
1]+b[7])))))))+6*exp(2*b[3]*(1/52))*b[4]*b[1]^3*b[6]*b[5]+6*exp((
(b[1]+3*b[3])).*(1/52))*b[3]^2*((b[2]*b[1]-b[10]*b[9]+b[8]*b[7])
).*((b[2]^2*b[1]^2+xt^2*b[1]^2-b[10]^2*b[1]*b[9]+b[10]^2*b[9]^2-
b[8]^2*b[1]*b[7]-2*b[10]*b[8]*b[9]*b[7]+b[8]^2*b[7]^2+2*xt*b[1]*
((b[10]*b[9]-b[8]*b[7]))-2*b[2]*b[1]*((xt*b[1]+b[10]*b[9]-b[8]*b
[7]))+b[4]*b[1]^2*(1/52)))+2*exp(3*b[3]*(1/52)).*(((-b[2]^3)*b[1
]^3*b[3]^2+xt^3*b[1]^3*b[3]^2+2*b[10]^3*b[1]^2*b[3]^2*b[9]-3*b[1
0]^3*b[1]*b[3]^2*b[9]^2+b[10]^3*b[3]^2*b[9]^3-2*b[8]^3*b[1]^2*b[
3]^2*b[7]+3*b[10]^2*b[8]*b[1]*b[3]^2*b[9]*b[7]-3*b[10]*b[8]^2*b[
1]*b[3]^2*b[9]*b[7]-3*b[10]^2*b[8]*b[3]^2*b[9]^2*b[7]+3*b[8]^3*b
[1]*b[3]^2*b[7]^2+3*b[10]*b[8]^2*b[3]^2*b[9]*b[7]^2-b[8]^3*b[3]^
2*b[7]^3+3*xt^2*b[1]^2*b[3]^2*((b[10]*b[9]-b[8]*b[7]))+3*b[2]^2*
b[1]^2*b[3]^2*((xt*b[1]+b[10]*b[9]-b[8]*b[7]))-3*b[4]*b[1]^3*b[6
]*b[5]+3*b[4]*b[10]*b[1]^2*b[3]^2*b[9]*(1/52)-3*b[4]*b[8]*b[1]^2
*b[3]^2*b[7]*(1/52)+3*b[4]*b[1]^3*b[3]*b[6]*b[5]*(1/52)+3*xt*b[1
]*b[3]^2*((b[10]^2*b[9]*(((-b[1])+b[9]))-2*b[10]*b[8]*b[9]*b[7]+
b[8]^2*b[7]*(((-b[1])+b[7]))+b[4]*b[1]^2*(1/52)))-3*b[2]*b[1]*b[
3]^2*((xt^2*b[1]^2+b[10]^2*b[9]*(((-b[1])+b[9]))-b[8]^2*b[1]*b[7
]-2*b[10]*b[8]*b[9]*b[7]+b[8]^2*b[7]^2+2*xt*b[1]*((b[10]*b[9]-b[
8]*b[7]))+b[4]*b[1]^2*(1/52)))))))));

m4=((1/b[1]^4).*((4^(-((b[4]*b[3])/b[5]^2))*exp((-((4*b[1]))).*(
1/52)).*((4^((b[4]*b[3])/b[5]^2).*(((((-1)+exp(b[1]*(1/52))))*b[
2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7
])))))^4+3*2^(1+(2*b[4]*b[3])/b[5]^2)*b[1]*(((((-1)+exp(b[1]*(1/5
2))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-
b[8]*b[7])))))^2*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1
)+exp(2*b[1]*(1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52)))+3*4^((b[4]
*b[3])/b[5]^2)*b[1]^2*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]
+(((-1)+exp(2*b[1]*(1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52)))^2+(1
/b[3]^2).*((4^(1+(b[4]*b[3])/b[5]^2)*exp((-b[3]).*(1/52))*b[1]^2
*(((((-1)+exp(b[1]*(1/52))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(
1/52)))).*((b[10]*b[9]-b[8]*b[7])))).*(((-2)*exp(((3*b[1]+b[3]))
.*(1/52))*b[3]^2*((b[10]^3*b[9]-b[8]^3*b[7]))+3*b[4]*b[1]*b[6]*b
[5]+exp(b[3]*(1/52)).*((2*b[10]^3*b[3]^2*b[9]-2*b[8]^3*b[3]^2*b[
7]+3*b[4]*b[1]*b[6]*b[5]*(((-1)+b[3]*(1/52)))))))))+(1/b[3]^3).*
((3*4^((b[4]*b[3])/b[5]^2)*exp((-b[3]).*(1/52))*b[1]^3*((2*exp((
(4*b[1]+b[3])).*(1/52))*b[3]^3*((b[10]^4*b[9]+b[8]^4*b[7]))+b[4]
*b[1]*b[5]^2*((1+4*b[6]^2*((2+b[3]*(1/52)))))-exp(b[3]*(1/52)).*
```

```
((2*b[10]^4*b[3]^3*b[9]+2*b[8]^4*b[3]^3*b[7]+b[4]*b[1]*b[5]^2*((
1-b[3]*(1/52)+b[6]^2*((8-4*b[3]*(1/52))))))))))))))));

m5=(((exp((-b[1]).*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[2]+
xt))-(((((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7])))))/(b[
1]*exp(b[1]*(1/52))))).*((exp((-b[1]).*(((1/52)+(1/52)))).*(((((
-1)+exp(b[1]*(((1/52)+(1/52))))))*b[2]+
xt))-(((((-1)+exp(b[1]*(((1/52)+(1/52))))))).*((b[10]*b[9]-b[8]*b
[7])))))/(b[1]*exp(b[1]*(((1/52)+(1/52)))))))-(exp((-b[1]).*(((1/
52)+2*(1/52)))).*(((-(((-1)+exp(2*b[1]*(1/52))))))*b[10]^2*b[9]-(
((-1)+exp(2*b[1]*(1/52))))*b[8]^2*b[7]-b[4]*b[1]*(1/52))))/b[1])
;

m6=(((exp((-b[1]).*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[2]+
xt))-(((((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7])))))/(b[
1]*exp(b[1]*(1/52))))).*((exp((-b[1]).*(((2/52)+(1/52)))).*(((((
-1)+exp(b[1]*(((2/52)+(1/52))))))*b[2]+
xt))-(((((-1)+exp(b[1]*(((2/52)+(1/52))))))).*((b[10]*b[9]-b[8]*b
[7])))))/(b[1]*exp(b[1]*(((2/52)+(1/52)))))))-(exp((-b[1]).*(((2/
52)+2*(1/52)))).*(((-(((-1)+exp(2*b[1]*(1/52))))))*b[10]^2*b[9]-(
((-1)+exp(2*b[1]*(1/52))))*b[8]^2*b[7]-b[4]*b[1]*(1/52))))/b[1])
;

m7=((-2)*exp((-((b[9]+b[7]))).*(((1/52)+(1/52)))).*((exp(1/52)))
^(b[9]+b[7]).*((exp(1/52)))^(b[9]+b[7]).*(((exp((-2)*b[1]*(((1/5
2)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3*b[9])/b[1]+(exp(
(-b[1]).*((2*(1/52)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3
*b[9])/b[1]+(exp((-2).*(1/52)*b[1]-3*b[1]*(1/52)).*(((-1)+exp(b[
1]*(1/52))))*b[10]^3*b[9])/b[1]-(exp((-2)*b[1]*(((1/52)+(1/52)))
).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp((-b[1]).*((2
*(1/52)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(
exp((-2).*(1/52)*b[1]-3*b[1]*(1/52)).*(((-1)+exp(b[1]*(1/52))))*
b[8]^3*b[7])/b[1]-(exp((-2).*(1/52)*b[1]-3*b[1]*(1/52)).*(((((-1
)+exp(b[1]*(1/52))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52))))
.*((b[10]*b[9]-b[8]*b[7])))).*(((((-1)+exp(b[1]*(((1/52)+(1/52))
))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(((1/52)+(1/52))))))).*((b
[10]*b[9]-b[8]*b[7])))))^2)./(2*b[1]^3)+(3*exp((-2).*(1/52)*b[1]-
((b[3]+3*b[1])).*(1/52))*b[4]*b[6]*b[5]*((1-exp(b[3]*(1/52))+b[3
]*(1/52)))))./(2*b[3]^2)-(exp((-2).*(1/52)*b[1]-3*b[1]*(1/52)).*(
((((-1)+exp(b[1]*(((1/52)+(1/52))))))*b[2]*b[1]+xt*b[1]-(((-1)+e
xp(b[1]*(((1/52)+(1/52))))))).*((b[10]*b[9]-b[8]*b[7])))).*(((((-
1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)+exp(2*b[1]*(1/52))))
*b[8]^2*b[7]+b[4]*b[1]*(1/52))))/b[1]^2-(exp((-2).*(1/52)*b[1]-3
*b[1]*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[2]*b[1]+xt*b[1]-(((
-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7])))).*((exp(2*b[1]
*(1/52)).*(1/52)*b[4]*b[1]-b[10]^2*b[9]-b[8]^2*b[7]+exp(2*b[1]*(
((1/52)+(1/52)))).*((b[10]^2*b[9]+b[8]^2*b[7]))+b[4]*b[1]*(1/52)
))).*(2*b[1]^2)-(3*exp((-2).*(1/52)*b[1]-((b[3]+3*b[1])).*(1/52)
)*b[4]*b[6]*b[5]*((2+b[3]*(1/52)+exp(b[3]*(1/52)).*(((-2)+b[3]*(
1/52))))))).*(2*b[3]^2)-(exp((-2).*(1/52)*b[1]-((b[3]+3*b[1])).*(
1/52))*b[4]*b[6]*b[5]*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52)))
)))/b[3]^2-(exp((-2).*(1/52)*b[1]-((b[3]+3*b[1])).*(1/52))*b[4]*
b[6]*b[5]*(1/52).*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))))))/
b[3]+(exp((-2).*(1/52)*b[1]-((b[3]+3*b[1])).*(1/52))*b[4]*b[6]*b
[5]*((1+b[3]*(1/52))).*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))
```

```
))))/b[3]^2)));

m8=((-2)*exp((-((b[9]+b[7]))).*(((1/52)+(1/52)))).*((exp(1/52)))
^(b[9]+b[7]).*((exp(1/52)))^(b[9]+b[7]).*(((exp((-b[1]).*(((1/52
)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3*b[9])/b[1]+(exp((
-b[1]).*(((1/52)+2*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3*
b[9])/b[1]+(exp((-b[1]).*(((1/52)+3*(1/52)))).*(((-1)+exp(b[1]*(
1/52))))*b[10]^3*b[9])/b[1]-(exp((-b[1]).*(((1/52)+(1/52)))).*((
(-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp((-b[1]).*(((1/52)
+2*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp((
-b[1]).*(((1/52)+3*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b
[7])/b[1]-(exp((-b[1]).*(((1/52)+3*(1/52)))).*(((((-1)+exp(b[1]*
(1/52))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*b
[9]-b[8]*b[7])))))^2
.*(((((-1)+exp(b[1]*(((1/52)+(1/52))))))*b[2]*b[1]+xt*b[1]-(((-1
)+exp(b[1]*(((1/52)+(1/52)))))).*((b[10]*b[9]-b[8]*b[7])))))./(2
*b[1]^3)+(3*exp((-b[3]).*(1/52)-b[1]*(((1/52)+3*(1/52))))*b[4]*b
[6]*b[5]*((1-exp(b[3]*(1/52))+b[3]*(1/52)))).*(2*b[3]^2)-(exp((-
b[1]).*(((1/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(1/52))))*b[2]*b[
1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7])))))
.*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)+exp(2*b[1]*(
1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52))))/b[1]^2-(exp((-b[1]).*((
(1/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(((1/52)+(1/52))))))*b[2]*
b[1]+xt*b[1]-(((-1)+exp(b[1]*(((1/52)+(1/52)))))).*((b[10]*b[9]-
b[8]*b[7])))).*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)
+exp(2*b[1]*(1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52)))).*(2*b[1]^2
)-(3*exp((-(1/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5
]*((2+b[3]*(1/52)+exp(b[3]*(1/52))).*(((-2)+b[3]*(1/52))))))./(2*
b[3]^2)-(exp((-(1/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]
*b[5]*((1+exp(b[3]*(1/52))).*(((-1)+b[3]*(1/52))))))/b[3]^2-(exp(
(-(1/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5]*(1/52).
*((1+exp(b[3]*(1/52))).*(((-1)+b[3]*(1/52))))))/b[3]+(exp((-(1/52
))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5]*((1+b[3]*(1/52
))).*((1+exp(b[3]*(1/52))).*(((-1)+b[3]*(1/52))))))/b[3]^2)));

m9=((-2)*exp((-((b[9]+b[7]))).*(((2/52)+(1/52)))).*((exp(2/52)))
^(b[9]+b[7]).*((exp(1/52)))^(b[9]+b[7]).*(((exp((-2)*b[1]*(((2/5
2)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3*b[9])/b[1]+(exp(
(-b[1]).*((2*(2/52)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3
*b[9])/b[1]+(exp((-2).*(2/52)*b[1]-3*b[1]*(1/52)).*(((-1)+exp(b[
1]*(1/52))))*b[10]^3*b[9])/b[1]-(exp((-2)*b[1]*(((2/52)+(1/52)))
).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp((-b[1]).*((2
*(2/52)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(
exp((-2).*(2/52)*b[1]-3*b[1]*(1/52)).*(((-1)+exp(b[1]*(1/52))))*
b[8]^3*b[7])/b[1]-(exp((-2).*(2/52)*b[1]-3*b[1]*(1/52)).*(((((-1
)+exp(b[1]*(1/52))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52))))
.*((b[10]*b[9]-b[8]*b[7])))).*(((((-1)+exp(b[1]*(((2/52)+(1/52))
))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(((2/52)+(1/52)))))).*((b
[10]*b[9]-b[8]*b[7])))))^2)./(2*b[1]^3)+(3*exp((-2).*(2/52)*b[1]-
((b[3]+3*b[1])).*(1/52))*b[4]*b[6]*b[5]*((1-exp(b[3]*(1/52))+b[3
]*(1/52)))).*(2*b[3]^2)-(exp((-2).*(2/52)*b[1]-3*b[1]*(1/52)).*(
((((-1)+exp(b[1]*(((2/52)+(1/52))))))*b[2]*b[1]+xt*b[1]-(((-1)+e
xp(b[1]*(((2/52)+(1/52)))))).*((b[10]*b[9]-b[8]*b[7])))).*(((((-
1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)+exp(2*b[1]*(1/52))))
*b[8]^2*b[7]+b[4]*b[1]*(1/52))))/b[1]^2-(exp((-2).*(2/52)*b[1]-3
```

```
*b[1]*(1/52)).*(((((-1)+exp(b[1]*(1/52))))*b[2]*b[1]+xt*b[1]-(((
-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7]))))).*((exp(2*b[1]
*(1/52)).*(2/52)*b[4]*b[1]-b[10]^2*b[9]-b[8]^2*b[7]+exp(2*b[1]*(
((2/52)+(1/52)))).*((b[10]^2*b[9]+b[8]^2*b[7]))+b[4]*b[1]*(1/52)
))).(2*b[1]^2)-(3*exp((-2).*(2/52)*b[1]-((b[3]+3*b[1])).*(1/52)
)*b[4]*b[6]*b[5]*((2+b[3]*(1/52)+exp(b[3]*(1/52)).*(((-2)+b[3]*(
1/52)))))./(2*b[3]^2)-(exp((-2).*(2/52)*b[1]-((b[3]+3*b[1])).*(
1/52))*b[4]*b[6]*b[5]*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52)))
)))/b[3]^2-(exp((-2).*(2/52)*b[1]-((b[3]+3*b[1])).*(1/52))*b[4]*
b[6]*b[5]*(1/52).*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))))))/
b[3]+(exp((-2).*(2/52)*b[1]-((b[3]+3*b[1])).*(1/52))*b[4]*b[6]*b
[5]*((1+b[3]*(1/52))).*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))
))))/b[3]^2)));

m10=((-2)*exp((-((b[9]+b[7]))).*(((2/52)+(1/52)))).*((exp(2/52))
)^(b[9]+b[7]).*((exp(1/52)))^(b[9]+b[7]).*(((exp((-b[1]).*(((2/5
2)+(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3*b[9])/b[1]+(exp(
(-b[1]).*(((2/52)+2*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[10]^3
*b[9])/b[1]+(exp((-b[1]).*(((2/52)+3*(1/52)))).*(((-1)+exp(b[1]*
(1/52))))*b[10]^3*b[9])/b[1]-(exp((-b[1]).*(((2/52)+(1/52)))).*(
((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp((-b[1]).*(((2/52
)+2*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*b[7])/b[1]-(exp(
(-b[1]).*(((2/52)+3*(1/52)))).*(((-1)+exp(b[1]*(1/52))))*b[8]^3*
b[7])/b[1]-(exp((-b[1]).*(((2/52)+3*(1/52)))).*(((((-1)+exp(b[1]
*(1/52))))*b[2]*b[1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*
b[9]-b[8]*b[7]))))^2
.*(((((-1)+exp(b[1]*(((2/52)+(1/52))))))*b[2]*b[1]+xt*b[1]-(((-1
)+exp(b[1]*(((2/52)+(1/52)))))).*((b[10]*b[9]-b[8]*b[7]))))))./(2
*b[1]^3)+(3*exp((-b[3]).*(1/52)-b[1]*(((2/52)+3*(1/52))))*b[4]*b
[6]*b[5]*((1-exp(b[3]*(1/52))+b[3]*(1/52))))./(2*b[3]^2)-(exp((-
b[1]).*(((2/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(1/52))))*b[2]*b[
1]+xt*b[1]-(((-1)+exp(b[1]*(1/52)))).*((b[10]*b[9]-b[8]*b[7]))))
.*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)+exp(2*b[1]*(
1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52))))/b[1]^2-(exp((-b[1]).*((
(2/52)+3*(1/52)))).*(((((-1)+exp(b[1]*(((2/52)+(1/52))))))*b[2]*
b[1]+xt*b[1]-(((-1)+exp(b[1]*(((2/52)+(1/52)))))).*((b[10]*b[9]-
b[8]*b[7])))).*(((((-1)+exp(2*b[1]*(1/52))))*b[10]^2*b[9]+(((-1)
+exp(2*b[1]*(1/52))))*b[8]^2*b[7]+b[4]*b[1]*(1/52)))))./(2*b[1]^2
)-(3*exp((-(2/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5
]*((2+b[3]*(1/52)+exp(b[3]*(1/52)).*(((-2)+b[3]*(1/52))))))./(2*
b[3]^2)-(exp((-(2/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]
*b[5]*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))))))/b[3]^2-(exp(
(-(2/52))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5]*(1/52).
*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))))))/b[3]+(exp((-(2/52
))*b[1]-b[3]*(1/52)-3*b[1]*(1/52))*b[4]*b[6]*b[5]*((1+b[3]*(1/52
))).*((1+exp(b[3]*(1/52)).*(((-1)+b[3]*(1/52))))))/b[3]^2)));
 moms=m1~m2~m3~m4~m5~m6~m7~m8~m9~m10;
retp(moms);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_SV: Returns the obj func for SV Model
                                   *
*****************************************************************
```

```
*********************************
* Inputs:       b      -   starting values
                                    *
* Output:              -   the objective function to be minimised
                                *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_SV(b);
   local
moms,g1,g2,g3,g4,g5,g6,gsubT,gsubTall,NWlags,NW,invNW,flagnw;
        moms=SV_moms(b);

   g1=meanc(xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]);
   g2=meanc(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2]);
   g3=meanc(xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]);
   g4=meanc(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4]);
   g5=meanc(xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5]);
   g6=meanc(xt[2:rows(xt)-2].*xt[4:rows(xt)]
-moms[1:rows(moms)-3,6]);
   gsubT=g1|g2|g3|g4|g5|g6;
   gsubTall=((xt[2:rows(xt)-2]-moms[1:rows(moms)-3,1]) ~
(xt[2:rows(xt)-2]^2-moms[1:rows(moms)-3,2])
            ~ (xt[2:rows(xt)-2]^3-moms[1:rows(moms)-3,3]) ~
(xt[2:rows(xt)-2]^4-moms[1:rows(moms)-3,4])
            ~ (xt[2:rows(xt)-2].*xt[3:rows(xt)-1]
-moms[1:rows(moms)-3,5])
        ~ (xt[2:rows(xt)-2].*xt[4:rows(xt)]
-moms[1:rows(moms)-3,6]) );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;

proc SV_moms(b);
local
moms,m1,m2,m3,m4,m5,m6,phi,c,d,Kapar,R_bar,Kapa_v,v_bar,Sigma_v,
Rho,iota,xi,r1,r2,u;

m1=exp((-b[1])*(1/52))*(((((-1)+exp(b[1]*(1/52))))*b[2]+xt));

m2=exp((-2)*b[1]*(1/52))*(((((-1)+exp(b[1]*(1/52))))^2*b[2]^2+2*
(((-1)+exp(b[1]*(1/52))))*b[2]*xt+xt^2+b[4]*(1/52)));

m3=(1/b[3]^2)*((exp((-3)*((2*b[1]+b[3]))*(1/52))*((exp(3*((2*b[1
]+b[3]))*(1/52))*b[2]^3*b[3]^2-3*exp(5*b[1]*(1/52)+3*b[3]*(1/52)
)*b[2]^2*((b[2]-xt))*b[3]^2+3*exp(3*b[1]*(1/52)+2*b[3]*(1/52))*b
[4]*b[6]*b[5]+3*exp(4*b[1]*(1/52)+3*b[3]*(1/52))*b[2]*b[3]^2*((b
```

```
[2]^2-2*b[2]*xt+xt^2+b[4]*(1/52)))+exp(3*((b[1]+b[3]))*(1/52))*(
((-b[2]^3)*b[3]^2+3*b[2]^2*xt*b[3]^2+xt^3*b[3]^2+3*xt*b[4]*b[3]^
2*(1/52)-3*b[2]*b[3]^2*((xt^2+b[4]*(1/52)))+3*b[4]*b[6]*b[5]*(((
-1)+b[3]*(1/52)))))))));

m4=(1/b[3]^3)*((exp((-4)*((2*b[1]+b[3]))*(1/52))*((exp(4*((2*b[1
]+b[3]))*(1/52))*b[2]^4*b[3]^3-4*exp(7*b[1]*(1/52)+4*b[3]*(1/52)
)*b[2]^3*((b[2]-xt))*b[3]^3+12*exp(5*b[1]*(1/52)+3*b[3]*(1/52))*
b[2]*b[4]*b[3]*b[6]*b[5]+6*exp(6*b[1]*(1/52)+4*b[3]*(1/52))*b[2]
^2*b[3]^3*((b[2]^2-2*b[2]*xt+xt^2+b[4]*(1/52)))+3*exp(4*b[1]*(1/
52)+3*b[3]*(1/52))*b[4]*b[5]*(((-4)*b[2]*b[3]*b[6]+4*xt*b[3]*b[6
]+b[5]+8*b[6]^2*b[5]+4*b[3]*b[6]^2*b[5]*(1/52)))-4*exp(5*b[1]*(1
/52)+4*b[3]*(1/52))*b[2]*b[3]*((b[2]^3*b[3]^2-3*b[2]^2*xt*b[3]^2
-xt^3*b[3]^2-3*xt*b[4]*b[3]^2*(1/52)+3*b[2]*b[3]^2*((xt^2+b[4]*(
1/52)))-3*b[4]*b[6]*b[5]*(((-1)+b[3]*(1/52)))))+exp(4*((b[1]+b[3
]))*(1/52))*((b[2]^4*b[3]^3-4*b[2]^3*xt*b[3]^3+xt^4*b[3]^3+6*xt^
2*b[4]*b[3]^3*(1/52)+6*b[2]^2*b[3]^3*((xt^2+b[4]*(1/52)))+12*xt*
b[4]*b[3]*b[6]*b[5]*(((-1)+b[3]*(1/52)))-4*b[2]*b[3]*((xt^3*b[3]
^2+3*xt*b[4]*b[3]^2*(1/52)+3*b[4]*b[6]*b[5]*(((-1)+b[3]*(1/52)))
))+3*b[4]*((b[4]*b[3]^3*(1/52)^2+b[5]^2*(((-1)+b[3]*(1/52)+4*b[6
]^2*(((-2)+b[3]*(1/52)))))))))))));

m5=(((((exp((-b[1])*(1/52))*(((((-1)+exp(b[1]*(1/52))))*b[2]+xt)
))).*((exp((-b[1])*(((1/52)+(1/52))))*(((((-1)+exp(b[1]*(((1/52)
+(1/52))))))*b[2]+xt))))+exp((-b[1])*((2*(1/52)+(1/52))))*b[4]*(
1/52))));

m6=(((((exp((-b[1])*(1/52))*(((((-1)+exp(b[1]*(1/52))))*b[2]+xt)
))).*((exp((-b[1])*(((1/52)+(2/52))))*(((((-1)+exp(b[1]*(((1/52)
+(2/52))))))*b[2]+xt))))+exp((-b[1])*((2*(1/52)+(2/52))))*b[4]*(
1/52))));
    moms=m1~m2~m3~m4~m5~m6;
retp(moms);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_SM: Returns the obj func for SM Model
                                  *
*************************************************************************
*******************************
* Inputs:       b      -   starting values
                                 *
* Output:           -   the objective function to be minimised
                                  *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_SM(b);
   local
moms,g1,g2,g3,g4,g5,gsubT,gsubTall,NWlags,NW,invNW,flagnw,gTall;
        moms=SM_moms(b);
   g1=meanc(xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]);
   g2=meanc(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2]);
   g3=meanc(xt[2:rows(xt)-1]^3-moms[1:rows(moms)-2,3]);
   g4=meanc(xt[2:rows(xt)-1]^4-moms[1:rows(moms)-2,4]);
```

```
   g5=meanc(xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,5]);
   gsubT=g1|g2|g3|g4|g5;
   gsubTall=((xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]) ~
(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2])
            ~ (xt[2:rows(xt)-1]^3-moms[1:rows(moms)-2,3]) ~
(xt[2:rows(xt)-1]^4-moms[1:rows(moms)-2,4])
            ~ (xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,5])  );
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*invNW*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;

proc SM_moms(b);
local moms,m1,m2,m3,m4,m5;
    m1=exp((-b[1]).*(1/52)).*((xt+b[1]*b[4].*(1/52)));

m2=(1/(2*b[3]^3*b[1])).*((exp((-((b[3]+2*b[1]))).*(1/52)).*((exp
(((b[3]+2*b[1])).*(1/52))*b[3]^3*b[2]^2+2*b[1]^3*b[4]*b[5]^2+exp
(b[3].*(1/52)).*(((-2)*b[1]^3*b[4]*b[5]^2+2*b[3]*b[1]^3*b[4]*b[5
]^2*(1/52)+b[3]^3*((2*xt^2*b[1]-b[2]^2+4*xt*b[1]^2*b[4].*(1/52)+
2*b[1]^3*b[4]^2*(1/52)^2))))))));

m3=(1/(2*b[3]^5*b[1])).*((exp((-((b[3]+3*b[1]))).*(1/52)).*((3*e
xp(((b[3]+2*b[1])).*(1/52))*b[3]^5*b[2]^2*((xt+b[1]*b[4].*(1/52)
))+6*b[1]^3*b[4]*b[5]^2*((2*b[1]*b[5]^2+b[3]*b[1]*b[5]^2*(1/52)+
b[3]^2*((xt+b[1]*b[4].*(1/52)))))
+exp(b[3].*(1/52)).*(((-12)*b[1]^4*b[4]*b[5]^4+6*b[3]*b[1]^4*b[4
]*b[5]^4*(1/52)-6*b[3]^2*b[1]^3*b[4]*b[5]^2*((xt
+b[1]*b[4].*(1/52)))+6*b[3]^3*b[1]^3*b[4]*b[5]^2*(1/52).*((xt+b[
1]*b[4].*(1/52)))+b[3]^5*((xt+b[1]*b[4].*(1/52))).*((2*xt^2*b[1]
-3*b[2]^2+4*xt*b[1]^2*b[4].*(1/52)+2*b[1]^3*b[4]^2*(1/52)^2)))))
)));

m4=(1/(4*b[3]^7*b[1]^2)).*((exp((-2).*((b[3]+2*b[1])).*(1/52)).*
((3*exp(2*((b[3]+2*b[1])).*(1/52))*b[3]^7*b[2]^4+12*exp(((b[3]+2
*b[1])).*(1/52))*b[3]^4*b[1]^3*b[4]*b[2]^2*b[5]^2+6*b[1]^6*b[4]*
b[5]^4*((2*b[3]*b[4]+b[5]^2))+6*exp(2*((b[3]+b[1])).*(1/52))*b[3
]^4*b[2]^2*(((-2)*b[1]^3*b[4]*b[5]^2+2*b[3]*b[1]^3*b[4]*b[5]^2*(
1/52)+b[3]^3*((2*xt^2*b[1]-b[2]^2+4*xt*b[1]^2*b[4].*(1/52)+2*b[1
]^3*b[4]^2*(1/52)^2))))+12*exp(b[3].*(1/52))*b[1]^3*b[4]*b[5]^2*
((14*b[1]^3*b[5]^4+4*b[3]^3*b[1]^2*b[5]^2*(1/52).*((xt+b[1]*b[4]
.*(1/52)))-2*b[3]*b[1]^3*b[5]^2*((b[4]-5*b[5]^2*(1/52)))+b[3]^4*
((2*xt^2*b[1]-b[2]^2+4*xt*b[1]^2*b[4].*(1/52)+2*b[1]^3*b[4]^2*(1
/52)^2))+2*b[3]^2*b[1]^2*b[5]^2*((4*xt+b[1].*(1/52).*((5*b[4]+b[
5]^2*(1/52)))))))
+exp(2*b[3].*(1/52)).*(((-174)*b[1]^6*b[4]*b[5]^6-24*b[3]^2*b[1]
```

```
^5*b[4]*b[5]^4*((4*xt+5*b[1]*b[4].*(1/52)))
+12*b[3]^3*b[1]^5*b[4]*b[5]^4*(1/52).*((4*xt+5*b[1]*b[4].*(1/52)
))+12*b[3]*b[1]^6*b[4]*b[5]^4*((b[4]+5*b[5]^2*(1/52)))-12*b[3]^4
*b[1]^3*b[4]*b[5]^2*((2*xt^2*b[1]-b[2]^2+4*xt*b[1]^2*b[4].*(1/52
)+2*b[1]^3*b[4]^2*(1/52)^2))+12*b[3]^5*b[1]^3*b[4]*b[5]^2*(1/52)
.*((2*xt^2*b[1]-b[2]^2+4*xt*b[1]^2*b[4].*(1/52)+2*b[1]^3*b[4]^2*
(1/52)^2))
+b[3]^7*((4*xt^4*b[1]^2+3*b[2]^4+16*xt^3*b[1]^3*b[4].*(1/52)-12*
b[1]^3*b[4]^2*b[2]^2*(1/52)^2+4*b[1]^6*b[4]^4*(1/52)^4+12*xt^2*(
((-b[1])*b[2]^2+2*b[1]^4*b[4]^2*(1/52)^2))+8*xt*(((-3)*b[1]^2*b[
4]*b[2]^2*(1/52)+2*b[1]^5*b[4]^3*(1/52)^3)))))))));

m5=((1/(2*((b[3]^2))^(3/2)*b[1]))*((exp((-((b[3]+b[1])))*(((1/52
)+2*(1/52))))*((exp(2*b[1]*(1/52)+b[3]*(((1/52)+2*(1/52))))*b[3]
^3*b[2]^2+2*exp(b[3]*(((1/52)+(1/52))))*b[1]^3*b[4]*b[5]^2+2*exp
(b[1]*(1/52)+b[3]*(((1/52)+2*(1/52))))*b[3]^3*(1/52)*b[1]^2*b[4]
*((xt+b[1]*b[4]*(1/52)))+exp(b[3]*(((1/52)+2*(1/52))))*(((-2)*b[
1]^3*b[4]*b[5]^2+2*b[3]*b[1]^3*b[4]*b[5]^2*(1/52)+b[3]^3*((2*xt^
2*b[1]-b[2]^2+4*xt*b[1]^2*b[4]*(1/52)+2*b[1]^3*b[4]^2*(1/52)^2))
)))))));
moms=m1~m2~m3~m4~m5;
retp(moms);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   GMM_CIR: Returns the obj func for CIR Model
                                  *
*****************************************************************
*********************************
* Inputs:       b      -   starting values
                                  *
* Output:           -   the objective function to be minimised
                                   *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc gmm_CIR(b);
   local
moms,g1,g2,g3,g4,g5,g6,gsubT,gsubTall,NWlags,NW,invNW,flagnw;
        moms=CIR_moms(b);

   g1=meanc(xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]);
   g2=meanc(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2]);
   g3=meanc(xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,3]);
   gsubT=g1|g2|g3;
   gsubTall=((xt[2:rows(xt)-1]-moms[1:rows(moms)-2,1]) ~
(xt[2:rows(xt)-1]^2-moms[1:rows(moms)-2,2])
            ~ (xt[2:rows(xt)-1].*xt[3:rows(xt)]
-moms[1:rows(moms)-2,3]));
   NWlags=int(rows(xt)^(1/6));
   NW=nwywest(gsubTall,NWlags);
   trap 1;
   invNW=inv(NW);
```

```
   flagnw = scalerr(invNW);
   if flagnw == 0;
       retp(gsubT'*inv(NW)*gsubT);
   else;
       retp(gsubT'*eye(rows(NW))*gsubT);
   endif;
endp;
proc CIR_moms(b);
local moms,m1,m2,m12;

m1=exp((-b[1])*(1/52))*(((((-1)+exp(b[1]*(1/52))))*b[2]+xt));

m2=(exp((-2)*b[1]*(1/52))*((2*b[1]*(((((-1)+exp(b[1]*(1/52))))*b
[2]+xt))^2+(((-1)+exp(b[1]*(1/52))))*(((((-1)+exp(b[1]*(1/52))))
*b[2]+2*xt))*b[3]^2)))/(2*b[1]);

m12=(exp((-3)*b[1]*(1/52))*((2*b[1]*(((((-1)+exp(b[1]*(1/52))))^
2*((1+exp(b[1]*(1/52))))*b[2]^2+(((-2)+exp(b[1]*(1/52))+exp(2*b[
1]*(1/52))))*b[2]*xt+xt^2))+(((-1)+exp(b[1]*(1/52))))*(((((-1)+e
xp(b[1]*(1/52))))*b[2]+2*xt))*b[3]^2)))/(2*b[1]);
        moms=m1~m2~m12;
retp(moms);
endp;




/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* ineq_One: One Factor Models Nonlinear Inequality constraint
                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc ineq_one(b);
retp(2*b[1]*b[2]-b[3]^2);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* ineq_two: Two Factor Models Nonlinear Inequality constraint
                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc ineq_two(b);
retp(2*b[3]*b[4]-b[5]^2);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* ineq_three: Three Factor Models Nonlinear Inequality
constraint                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc ineq_three(b);
retp((2*b[2]*b[3]-b[4]^2)|(2*b[5]*b[6]-b[7]^2));
endp;

proc Nwywest(hhat,pp);
  local N,df,m,Omega0,jj,Shat,Omegaj,titl,flag1;
  hhat = hhat - meanc(hhat)';
  N = rows(hhat);
  m = minc( pp|(N-2) );
  Omega0 = hhat'hhat/N;
    jj = 1;
  Shat  = Omega0;
  do until jj > pp;
    Omegaj = hhat[jj+1:N,.]'hhat[1:N-jj,.]/N;
    Shat   = Shat  +  ( 1 - jj/(m+1) ) * (Omegaj + Omegaj');
    jj = jj + 1;
  endo;
  retp(Shat);
endp;



/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* DGP_CIR:   Data Generation as CIR process
                               *
*           dp(t)=phi*(p_bar-p(t))*dt+sig*sqrt(p(t)*dW(t)
                               *
****************************************************************
*********************************
* Inputs:       T       -   Length of time series
                                   *
*                   h       -   discretization interval
                                       *
*                   phi     -   mean reversion parameter Process
                                       *
*                   p_bar   -   mean level
                                       *
*                   sig1    -   variance term
                                       *
*                   see     -   seed of rndns
                       *
* Output:   dat     -   time series
                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc DGP_CIR(T,h,phi,p_bar,sig1,see);
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=p_bar;// here we use the estimated parameter as start
value
        i=0;
```

```
        do while i < TN;
                i=i+1;

X1=Pt+phi*(p_bar-Pt)*h+sig1*sqrt(pt)*ee[i]
            -0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*h
            +0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*(ee[i]^2);
        /*Theory implies that this condition will never be
reached as when process approaches
        zero the volatility is switched off, that result depends
on h going to zero here h is
        very small indeed I am including the condition as a
final safe guard, here I am switching
        off the volatility manually something that should
happens assymptotically*/
        if X1 < 0;
            X1=Pt+phi*(p_bar-Pt)*h; print " same
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$";
        endif;
                if i%(h^-1)==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* DGP_CIRsim:   Data Generation as CIR process for simulation,
             The only difference with above is that we are given
the start value of X(0)
          *
*           dp(t)=phi*(p_bar-p(t))*dt+sig*sqrt(p(t)*dW(t)
                                      *
****************************************************************
*********************************/

proc DGP_CIRsim(T,h,phi,p_bar,sig1,see,start);
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=start; // here we use the given X(0) as start value
        i=0;
        do while i < TN;
                i=i+1;
                X1=Pt+phi*(p_bar-Pt)*h+sig1*sqrt(pt)*ee[i]
            -0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*h
            +0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*(ee[i]^2);
        /*Theory implies that this condition will never be
reached as when process approaches
        zero the volatility is switched off, that result depends
on h going to zero here h is
        very small indeed I am including the condition as a
final safe guard, here I am switching
```

```
        off the volatility manually something that should
happens assymptotically*/
        if X1 < 0;
            X1=Pt+phi*(p_bar-Pt)*h;
        endif;
                if i%((h^-1))==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                               *
* DGP: SM
                               *
*        dr(t)      =   kapa_r*(theta(t)-r(t))*dt+sig1*dW1(t)
                               *
*        dTheta(t)  =
kapa_Theta*(theta_bar-theta(t))*dt+(sig3*sqrt(theta(t)))*dW3(t)
       *
*        W's are mulually independent Brownian motions
                               *
****************************************************************
*********************************
* Inputs:       T: Length of time series
                                   *
*                   h: discretization interval
                                       *
*                   kapa_r: mean reversion parameter Interest
Rate Process                              *
*                   sig1: variance term interest rate Process
                                       *
*                   kapa_theta: mean reversion parameter mean
process                                   *
*                   theta_bar: mean  Level, also used as the
starting value of the mean process        *
*                   sig3: variance term mean Process
                                       *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(2)= DGP_SM(T,h,kapa_r,sig1,kapa_theta,theta_bar,sig3,see);
local TN,e1,e2,dat,Pt,Mt,i,zi1,zi2,r,r2,rhop,I21,M1,XX,test;
    test={};
        dat={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        Pt=theta_bar;
    Mt=theta_bar;
        i=0;
```

```
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sig1*e1[i];
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e2[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|M1;
                endif;
                Pt=XX;
        Mt=M1;
        endo;
retp(dat,test);
endp;




proc(2)=
DGP_SMsim(T,h,kapa_r,sig1,kapa_theta,theta_bar,sig3,see,start1,s
tart2);
local TN,e1,e2,dat,Pt,Mt,i,zi1,zi2,r,r2,rhop,I21,M1,XX,test;
    test={};
        dat={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        Pt=start1; // here we use the given X(0) as start value
    Mt=start2; // here we use the given seta(0) as start value
        i=0;
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sig1*e1[i];
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e2[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|M1;
                endif;
                Pt=XX;
        Mt=M1;
        endo;
retp(dat,test);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                                                              *
* DGP:SV
                                                              *
```

```
*        dr(t)  =   kapa_r*(r_bar-r(t))*dt+(sqrt(V(t)))*dW1(t)
                                  *
*        dV(t)  =
kapa_v*(v_bar-V(t))*dt+(sig2*sqrt(V(t)))*dW2(t)
            *
*****************************************************************
*******************************
* Inputs:       T: Length of time series
                                    *
*                   h: discretization interval
                                     *
*                   kapa_r: mean reversion parameter Interest
Rate Process
                                                *
*                   r_bar: mean  Level, also used as the
starting value of the mean process              *
*                   kapa_v: mean reversion parameter volatility
process
                                     *
*                   v_bar: mean Volatility Level, also used as
the starting value of the vol process    *
*                   sig2: variance term Volatility Process
                                     *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(2)= DGP_SV(T,h,kapa_r,r_bar,kapa_v,v_bar,sig2,rho,see);
local p,TN,e1,e2,dat,Pt,Vt,i,zi1,zi2,r,r2,rhop,I21,V1,XX,test;
    test={};
    p=1/h;
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=r_bar;
    Vt=v_bar;
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(r_bar-Pt)*h+sqrt(Vt)*(e1[i]*((1-rho^2)^0.5)+rho*e2
[i]);
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|v1;
                endif;
                Pt=XX;
        Vt=V1;
        endo;
retp(dat,test);
endp;
```

```
proc(2)=
DGP_SVsim(T,h,kapa_r,r_bar,kapa_v,v_bar,sig2,rho,see,start1,star
t2);
local p,TN,e1,e2,dat,Pt,Vt,i,zi1,zi2,r,r2,rhop,I21,V1,XX,test;
    test={};
    p=1/h;
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=start1;
    Vt=start2;
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(r_bar-Pt)*h+sqrt(Vt)*(e1[i]*((1-rho^2)^0.5)+rho*e2
[i]);
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|v1;
                endif;
                Pt=XX;
        Vt=V1;
        endo;
retp(dat,test);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                                  *
* DGP:SVj
                                   *
*****************************************************************
*******************************
* Inputs:       T: Length of time series
                                        *
*                   h: discretization interval
                                          *
*                   kapa_r: mean reversion parameter Interest
Rate Process
                                                *
*                   r_bar: mean  Level, also used as the
starting value of the mean process                  *
*                   kapa_v: mean reversion parameter volatility
process
                                          *
*                   v_bar: mean Volatility Level, also used as
the starting value of the vol process    *
*                   sig2: variance term Volatility Process
                                          *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(2)=
DGP_SVJ(T,h,kapa_r,r_bar,kapa_v,v_bar,sig2,rho,lamdau,NUu,lamdad
,NUd,see);
local
p,TN,e1,e2,dat,Pt,Vt,i,zi1,zi2,r,r2,rhop,I21,V1,XX,test,e3,jumpu
,p_jumpu,jumpd,p_jumpd,jump;
    test={};
    p=1/h;
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        e3 = rndus(TN,4,see);
    jumpu=-NUu*ln(e3[.,2]);
    p_jumpu=exp(-lamdau*h);
    p_jumpu=e3[.,1].>p_jumpu;
    jumpu=jumpu.*p_jumpu;

    jumpd=-NUd*ln(e3[.,4]);
    p_jumpd=exp(-lamdad*h);
    p_jumpd=e3[.,3].>p_jumpd;
    jumpd=jumpd.*p_jumpd;
    jump=jumpu-jumpd;
        dat={};
        Pt=r_bar;
    Vt=v_bar;
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(r_bar-Pt)*h+sqrt(Vt)*(e1[i]*((1-rho^2)^0.5)+rho*e2
[i])+jump[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        /*Theory implies that this condition will never be
reached as when process approaches
        zero the volatility is switched off, that result depends
on h going to zero. Here h
        is very small indeed, I am including the condition as a
final safe-guard, here I am
        switching off the volatility manually as a precaution
something that should happens
        assymptotically*/
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|v1;
                endif;
                Pt=XX;
        Vt=V1;
        endo;
retp(dat,test);
endp;
```

```
proc(2)=
DGP_SVJsim(T,h,kapa_r,r_bar,kapa_v,v_bar,sig2,rho,lamdau,NUu,lam
dad,NUd,see,start1,start2);
local
p,TN,e1,e2,dat,Pt,Vt,i,zi1,zi2,r,r2,rhop,I21,V1,XX,test,e3,jumpu
,p_jumpu,jumpd,p_jumpd,jump;
    test={};
    p=1/h;
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);
        e2= sqrt(h)*rndns(TN,1,see);
        e3 = rndus(TN,4,see);
    jumpu=-NUu*ln(e3[.,2]);
    p_jumpu=exp(-lamdau*h);
    p_jumpu=e3[.,1].>p_jumpu;
    jumpu=jumpu.*p_jumpu;

    jumpd=-NUd*ln(e3[.,4]);
    p_jumpd=exp(-lamdad*h);
    p_jumpd=e3[.,3].>p_jumpd;
    jumpd=jumpd.*p_jumpd;
    jump=jumpu-jumpd; //calculate Jump;
        dat={};
        Pt=start1;
    Vt=start2;
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(r_bar-Pt)*h+sqrt(Vt)*(e1[i]*((1-rho^2)^0.5)+rho*e2
[i])+jump[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        /*Theory implies that this condition will never be
reached as when process approaches
        zero the volatility is switched off, that result depends
on h going to zero. Here h
        is very small indeed, I am including the condition as a
final safe-guard, here I am
        switching off the volatility manually as a precaution
something that should happens
        assymptotically*/
                if i%(h^-1)==0;
                        dat=dat|XX;
                test=test|v1;
                endif;
                Pt=XX;
        Vt=V1;
        endo;
retp(dat,test);
```

```
endp;
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                               *
* DGP: CHEN
                               *
*        dr(t)      =
kapa_r*(theta(t)-r(t))*dt+(sqrt(V(t)))*dW1(t)
        *
*        dV(t)      =
kapa_v*(v_bar-V(t))*dt+(sig2*sqrt(V(t)))*dW2(t)
        *
*        dTheta(t)  =
kapa_Theta*(theta_bar-theta(t))*dt+(sig3*sqrt(theta(t)))*dW3(t)
        *
*        W's are mulually independent Brownian motions
                               *
*******************************************************************
*********************************
* Inputs:       T: Length of time series
                                   *
*                   h: discretization interval
                                       *
*                   kapa_r: mean reversion parameter Interest
Rate Process                              *
*                   kapa_v: mean reversion parameter volatility
process                                 *
*                   v_bar: mean Volatility Level, also used as
the starting value of the vol process    *
*                   sig2: variance term Volatility Process
                                       *
*                   kapa_theta: mean reversion parameter mean
process                                   *
*                   theta_bar: mean  Level, also used as the
starting value of the mean process         *
*                   sig3: variance term mean Process
                                       *
*                   sim_ind: indicator for generating the errors
                                       *
*                   sim_ind = 0 if errors are to be generated
randomly                              *
*                   sim_ind = 1 if errors are to be be taken
from a global variable (SGMM)      *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(3)=
DGP_chen(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,sig3,
see);
local theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);           e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};
```

```
        Pt=theta_bar;    Vt=v_bar;    Mt=theta_bar;
        i=0;
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sqrt(Vt)*e1[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
                Pt=XX;        Vt=V1;        Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;



proc(3)=
DGP_chensim(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,si
g3,see,start1,start2,start3);
local theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);            e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=start1;    Vt=start2;    Mt=start3;// change start
value for diffusion process
        i=0;
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sqrt(Vt)*e1[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
                Pt=XX;        Vt=V1;        Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;
```

```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                               *
* DGP: feed
                                 *
*****************************************************************
*********************************
* Inputs:       T: Length of time series
                                   *
*                   h: discretization interval
                                       *
*                   kapa_r: mean reversion parameter Interest
Rate Process                              *
*                   kapa_r_v: parameter capturig the effect of
volatility on conditional mean           *
*                   rho: conditional correlation between short
rate and its stochastic volatility       *
*                   kapa_v: mean reversion parameter volatility
process                                 *
*                   v_bar: mean Volatility Level, also used as
the starting value of the vol process    *
*                   sig2: variance term Volatility Process
                                       *
*                   kapa_theta: mean reversion parameter mean
process                                   *
*                   kapa_theta_v: parameter capturig the effect
of volatility on mean theta             *
*                   theta_bar: mean  Level, also used as the
starting value of the mean process         *
*                   sig3: variance term mean Process
                                       *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(3)=
DGP_Feed(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,sig3,
kapa_r_v,rho,see);
local theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);            e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=theta_bar;    Vt=v_bar;    Mt=theta_bar;
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(Mt-Pt)*h+kapa_r_v*(v_bar-Vt)*h+sqrt(Vt)*(e1[i]*((1
-rho^2)^0.5)+rho*e2[i]);
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
```

```
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
                Pt=XX;        Vt=V1;        Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;




proc(3)=
DGP_Feedsim(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,si
g3,kapa_r_v,rho,see,start1,start2,start3);
local theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);            e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};
        Pt=start1;    Vt=start2;    Mt=start3;// change start
value for diffusion process
        i=0;
        do while i < TN;
                i=i+1;

XX=Pt+kapa_r*(Mt-Pt)*h+kapa_r_v*(v_bar-Vt)*h+sqrt(Vt)*(e1[i]*((1
-rho^2)^0.5)+rho*e2[i]);
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
                Pt=XX;        Vt=V1;        Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data Generation
                              *
* DGP: CHEN_JUMP
                              *
```

```
*****************************************************************
*********************************
* Inputs:       T: Length of time series
                                                   *
*                   h: discretization interval
                                                       *
*                   kapa_r: mean reversion parameter Interest
Rate Process                                              *
*                   kapa_v: mean reversion parameter volatility
process                                                *
*                   v_bar: mean Volatility Level, also used as
the starting value of the vol process    *
*                   sig2: variance term Volatility Process
                                                       *
*                   kapa_theta: mean reversion parameter mean
process                                                   *
*                   theta_bar: mean  Level, also used as the
starting value of the mean process         *
*                   sig3: variance term mean Process
                                                       *
*                   sim_ind: indicator for generating the errors
                                                       *
*                   sim_ind = 0 if errors are to be generated
randomly                                         *
*                   sim_ind = 1 if errors are to be be taken
from a global variable (SGMM)        *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc(3)=
DGP_chenJ(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,sig3
,lamdau,NUu,lamdad,NUd,see);
local
theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1,e4,jumpu,p
_jumpu,jumpd,p_jumpd,jump;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);            e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};  e4 = rndus(TN,4,see);
    jumpu=-NUu*ln(e4[.,2]);
    p_jumpu=exp(-lamdau*h);
    p_jumpu=e4[.,1].>p_jumpu;
    jumpu=jumpu.*p_jumpu;
    jumpd=-NUd*ln(e4[.,4]);
    p_jumpd=exp(-lamdad*h);
    p_jumpd=e4[.,3].>p_jumpd;
    jumpd=jumpd.*p_jumpd;
    jump=jumpu-jumpd;
        Pt=theta_bar;    Vt=v_bar;    Mt=theta_bar;
        i=0;
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sqrt(Vt)*e1[i]+jump[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
```

```
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
                Pt=XX;        Vt=V1;        Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;



proc(3)=
DGP_chenJsim(T,h,kapa_r,kapa_v,v_bar,sig2,kapa_theta,theta_bar,s
ig3,lamdau,NUu,lamdad,NUd,see,start1,start2,start3);
local
theta_fact,v_fact,TN,e1,e2,e3,dat,pt,vt,mt,i,xx,v1,m1,e4,jumpu,p
_jumpu,jumpd,p_jumpd,jump;
    v_fact={};theta_fact={};
        TN=round(T/h);
        e1= sqrt(h)*rndns(TN,1,see);            e2=
sqrt(h)*rndns(TN,1,see);        e3= sqrt(h)*rndns(TN,1,see);
        dat={};  e4 = rndus(TN,4,see);
    jumpu=-NUu*ln(e4[.,2]);
    p_jumpu=exp(-lamdau*h);
    p_jumpu=e4[.,1].>p_jumpu;
    jumpu=jumpu.*p_jumpu;
    jumpd=-NUd*ln(e4[.,4]);
    p_jumpd=exp(-lamdad*h);
    p_jumpd=e4[.,3].>p_jumpd;
    jumpd=jumpd.*p_jumpd;
    jump=jumpu-jumpd;
        Pt=start1;    Vt=start2;    Mt=start3;// change start
value for diffusion process
        i=0;
        do while i < TN;
                i=i+1;
        XX=Pt+kapa_r*(Mt-Pt)*h+sqrt(Vt)*e1[i]+jump[i];
        V1=Vt+(kapa_v*(v_bar-Vt))*h+sig2*sqrt(vt)*e2[i];
            if V1 < 0;
            V1=Vt+(kapa_v*(v_bar-Vt))*h;
        endif;
        M1=Mt+(kapa_theta*(theta_bar-Mt))*h+sig3*sqrt(Mt)*e3[i];
            if M1 < 0;
            M1=Mt+(kapa_theta*(theta_bar-Mt))*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|XX;        v_fact=v_fact|v1;
    theta_fact=theta_fact|M1;
                endif;
```

```
                Pt=XX;          Vt=V1;          Mt=m1;
        endo;
retp(dat,v_fact,theta_fact);
endp;




/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* Data statistics
                               *

             *
****************************************************************
*********************************/

proc dis_stat(yt);
local avg,std,skew,kurt,stat;
    avg=meanc(yt);
    std=stdc(yt);
    skew=meanc(((yt-meanc(yt))/stdc(yt))^3);
    kurt=meanc(((yt-meanc(yt))/stdc(yt))^4);
    stat=avg|std|skew|kurt;
retp(stat);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* estimate: estimate the parameters
                    *
****************************************************************
*********************************
* Inputs:       dat1    -   time series
                                   *
*                   indicator  -   model indicator
                                       *
* Output:   est     -   estimated parameters
                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
/*
xt: is the lagged xt
model_ind; model indicator
    = 1 CIR
    = 2 SV
    = 3 SVJ :
    = 4 CHEN
    = 5 CHEN_JUMP
    = 6 SM
*/

proc(1)=estimate(xt,model_ind);
local b_start,b,f,g,retcode;
b={};
```

```
  if model_ind==1;
    coset;
    __output=0;
    _co_Algorithm=3;
    _co_LineSearch=4;
    _co_C = {  1 0 0,
               0 1 0,
               0 0 1};
    _co_D = {  0,
               0,
               0};
    _co_IneqProc = &ineq_one;
    b_start=0.1701|0.05|0.0249;
    {b,f,g,retcode} = co(&GMM_cir,b_start);


elseif model_ind==2;
    coset;
    __output=0;
    _co_Algorithm=3;
    _co_LineSearch=4;
    _co_DirTol=1e-4;
    _co_C = {1 0 0 0 0 0,
             0 1 0 0 0 0,
             0 0 1 0 0 0,
             0 0 0 1 0 0,
             0 0 0 0 1 0,
             0 0 0 0 0 1,
             0 0 0 0 0 -1};
    _co_D = {0,
             0,
             0,
             0,
             0,
            -1,
             0};
    _co_IneqProc = &ineq_two;
    b_start=0.27|0.05|3.5|0.00006|0.02|-0.5;
    {b,f,g,retcode} = co(&GMM_SV,b_start);

elseif model_ind==3;
    coset;
    __output=0;
    _co_Algorithm=3;
    _co_LineSearch=1;
    _co_DirTol=1e-4;
    _co_TrustRadius=1;
    _co_IneqProc = &ineq_two;
    _co_A = {0 0 0 0 0 0 1 0 0 0,
             0 0 0 0 0 0 0 1 0 0,
             0 0 0 0 0 0 0 0 1 0,
             0 0 0 0 0 0 0 0 0 1};
    _co_B = {5.4979,
             0.0006,
```

```
                2.121,
                0.002};// add equality constraint for this one,
becoz H fails, fix lamda in SVJ estimation
        _co_C = {    1 0 0 0 0 0 0 0 0 0,
                     0 1 0 0 0 0 0 0 0 0,
                     0 0 1 0 0 0 0 0 0 0,
                     0 0 0 1 0 0 0 0 0 0,
                     0 0 0 0 1 0 0 0 0 0,
                     0 0 0 0 0 1 0 0 0 0,
                     0 0 0 0 0 -1 0 0 0 0};
        _co_D = {0,0,0,0,0,
                 -1,
                 0};

b_start=0.230503|0.052496|2.292159|0.000019|0.0100042|-0.105|7.8
99973|0.001056|1.600042|0.001922;
       {b,f,g,retcode} = co(&gmm_SVJ,b_start);
//print "f=" f;

    elseif model_ind==4;
        coset;
        __output=0;
        _co_Algorithm=3;
        _co_LineSearch=1;
        _co_Trust=1;
        _co_IneqProc = &ineq_three;
        b_start=0.2|3|0.0003|0.01|0.3|0.06|0.04;
        _co_A = {0 0 0 0 1 0 0 };
        _co_B = {0.29};// add equality constraint for this one,
becoz H fails
        _co_C = {1 0 0 0 0 0 0,
                 0 1 0 0 0 0 0,
                 0 0 1 0 0 0 0,
                 0 0 0 1 0 0 0,
                 0 0 0 0 1 0 0,
                 0 0 0 0 0 1 0,
                 0 0 0 0 0 0 1};
        _co_D = {0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0};
       {b,f,g,retcode} = co(&gmm_C,b_start);
//print "f=" f;

    elseif model_ind==5;
          coset;
        __output=0;
        _co_Algorithm=3;
        _co_LineSearch=1;
        _co_Trust=1;
       _co_IneqProc = &ineq_three;
        _co_DirTol=1e-4;
```

```
b_start=0.363825|2.119171|0.000465|0.00555|0.287034|0.060095|0.1
84442|8.106309|0.000205|2.828375|0.000471;
       _co_C = {1 0 0 0 0 0 0 0 0 0 0,
                 0 1 0 0 0 0 0 0 0 0 0,
                 0 0 1 0 0 0 0 0 0 0 0,
                 0 0 0 1 0 0 0 0 0 0 0,
                 0 0 0 0 1 0 0 0 0 0 0,
                 0 0 0 0 0 1 0 0 0 0 0,
                 0 0 0 0 0 0 1 0 0 0 0,
                 0 0 0 0 0 0 0 1 0 0 0,
                 0 0 0 0 0 0 0 0 1 0 0,
                 0 0 0 0 0 0 0 0 0 1 0,
                 0 0 0 0 0 0 0 0 0 0 1};
        _co_D = {0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0,
                 0};
       {b,f,g,retcode} = co(&gmm_CJ,b_start);

    elseif model_ind==6;
        coset;
        __output=0;
        _co_A = {1 0 0 0 0};
        _co_B = {0.5};// add equality constraint for this one,
becoz H fails
        _co_C = {1 0 0 0 0,
                 0 1 0 0 0,
                 0 0 1 0 0,
                 0 0 0 1 0,
                 0 0 0 0 1};
        _co_D = {0,
                 0,
                 0,
                 0,
                 0};
        _co_IneqProc = &ineq_two;
        b_start=1.8|0.03|0.3|0.06|0.04;
        {b,f,g,retcode} = co(&GMM_SM,b_start);
//print " f=" f;

  endif;

retp(b);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS_stat: Calculates The Test Statistics
                                                    *
****************************************************************
*************************************************
* Inputs:       xxt      -      time series
            R                The first R observations that used
for estimation
       *
*                   tao     -          a vector indicating the
different step ahead con. int. to be examined
*
*                   S       -          number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV
  *
*                   mod_ind1 -         model specification 1

*                   mod_ind2 -         model specification 2
                                  *
*                   h       -          discretization interval

*
*                   u_bar   -          confidence interval

   *
* Output:
*           vt  -       Test statistics will have rows(tao)
results
            v1          First part of Test statistics
            v2          Second part of Test statistics
                                            *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)= CS_stat(xxt,R,N,tao,S,mod_ind1,mod_ind2,h,u_bar);
local
T,tt,xp,p_true,b,f,g,retcode,vt,p_sim1,p_sim2,i,sup_vt,b1,b2,see
,v1,v2;
    see=12343;//dont change seed, to use the same random
error,but different start value of X(t)
    T=rows(xxt);
    vt=zeros(rows(tao),1);
    v1=zeros(rows(tao),1);
    v2=zeros(rows(tao),1);

    i=0;
    do while i <rows(tao);
        i=i+1;
        tt=R;// the start of out-of-sample
        do while tt<=T-tao[i];
            xp=xxt[tt+tao[i],.];// the actual value of X(P), tt
is the in-sample obs
            p_true=(xp .> u_bar[1,1]).*(xp .< u_bar[2,1]);
            xt=xxt[1:tt,.];//set xt
```

```
p_sim1=Con_den(xt,tao[i],S,N,mod_ind1,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den(xt,tao[i],S,N,mod_ind2,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))
            v1[i]=v1[i]+(p_sim1-p_true).^2;
            v2[i]=v2[i]+(p_sim2-p_true).^2;
            vt[i]=v1[i]-v2[i];

   // print " tt,p_sim1~p_sim2~p_true~v1~v2~vt[i]=="
tt~p_sim1~p_sim2~p_true~v1[i]~v2[i]~vt[i];
            tt=tt+1;
        endo;
        v1[i]=v1[i]./sqrt(T-R-tao[i]+1);
        v2[i]=v2[i]./sqrt(T-R-tao[i]+1);
        vt[i]=vt[i]./sqrt(T-R-tao[i]+1);

    endo;
retp(vt~v1~v2);
endp;



/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS_stat2: Calculates The Test Statistics : Alternative method,
which need a latent variable series as
            input for the simulation process.
                                                           *
***************************************************************
*************************************************
* Inputs:       xxt      -      time series
            R               The first R observations that used
for estimation
       *
*                   tao     -           a vector indicating the
different step ahead con. int. to be examined
*
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV
  *
*                   model_ind1 -        model specification 1

*                   model_ind2 -        model specification 2
                                *
*                   h       -           discretization interval

*
*                   u_bar   -           confidence interval

            svSim1,mvSim1,svSim2,mvSim2 - latent variables
                                                           *
```

```
* Output:
*           vt  -       Test statistics will have rows(tao)
results                                            *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)=
CS_stat2(xxt,R,N,tao,S,model_ind1,model_ind2,h,u_bar,svSim1,mvSi
m1,svSim2,mvSim2);
local
T,tt,xp,p_true,b,f,g,retcode,vt,p_sim1,p_sim2,i,sup_vt,b1,b2,x_s
im1,x_sim2,see,v1,v2;
    see=12343;//dont change seed, to use the same random
error,but different start value of X(t)
    T=rows(xxt);

    vt=zeros(rows(tao),1);
    v1=zeros(rows(tao),1);
    v2=zeros(rows(tao),1);
    i=0;
    do while i <rows(tao);
        i=i+1;
        tt=R;// the start of out-of-sample
        do while tt<=T-tao[i];
            xp=xxt[tt+tao[i],.];// the actual value of X(P), tt
is the in-sample obs
            p_true=(xp .> u_bar[1,1]).*(xp .< u_bar[2,1]);
            xt=xxt[1:tt,.];

p_sim1=Con_den2(xt,tao[i],S,N,model_ind1,h,u_bar,see,svSim1,mvSi
m1);// prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den2(xt,tao[i],S,N,model_ind2,h,u_bar,see,svSim2,mvSi
m2);// prob(u1<x_sim(t+tao)<u2|x(t))
            v1[i]=v1[i]+(p_sim1-p_true).^2;
            v2[i]=v2[i]+(p_sim2-p_true).^2;
            vt[i]=v1[i]-v2[i];
//print " tt,p_sim1~p_sim2~p_true~v1~v2~vt[i]=="
tt~p_sim1~p_sim2~p_true~v1[i]~v2[i]~vt[i];
            tt=tt+1;
        endo;
        v1[i]=v1[i]./sqrt(T-R-tao[i]+1);
        v2[i]=v2[i]./sqrt(T-R-tao[i]+1);
        vt[i]=vt[i]./sqrt(T-R-tao[i]+1);

    endo;
retp(vt~v1~v2);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS_stat3: Calculates The Test Statistics : Alternative method,
which dont simulate latent variables
                    *
****************************************************************
```

```
*************************************************
* Inputs:       xxt      -      time series
            R               The first R observations that used
for estimation
       *
*                   tao     -           a vector indicating the
different step ahead con. int. to be examined
*
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV
  *
*                   model_ind1 -        model specification 1

*                   model_ind2 -        model specification 2
                                *
*                   h       -           discretization interval

*
*                   u_bar   -           confidence interval

                                                                       *
* Output:
*           vt  -       Test statistics will have rows(tao)
results                                            *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)= CS_stat3(xxt,R,N,tao,S,model_ind1,model_ind2,h,u_bar);
local
T,tt,xp,p_true,b,f,g,retcode,vt,p_sim1,p_sim2,i,sup_vt,b1,b2,x_s
im1,x_sim2,see,v1,v2;
    see=12343;//dont change seed, to use the same random
error,but different start value of X(t)
    T=rows(xxt);

    vt=zeros(rows(tao),1);
    v1=zeros(rows(tao),1);
    v2=zeros(rows(tao),1);
    i=0;
    do while i <rows(tao);
        i=i+1;
        tt=R;// the start of out-of-sample
        do while tt<=T-tao[i];
            xp=xxt[tt+tao[i],.];// the actual value of X(P), tt
is the in-sample obs
            p_true=(xp .> u_bar[1,1]).*(xp .< u_bar[2,1]);
            xt=xxt[1:tt,.];

p_sim1=Con_den3(xt,tao[i],S,N,model_ind1,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den3(xt,tao[i],S,N,model_ind2,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))
            v1[i]=v1[i]+(p_sim1-p_true).^2;
```

```
            v2[i]=v2[i]+(p_sim2-p_true).^2;
            vt[i]=v1[i]-v2[i];
//print " tt,p_sim1~p_sim2~p_true~v1~v2~vt[i]=="
tt~p_sim1~p_sim2~p_true~v1[i]~v2[i]~vt[i];
            tt=tt+1;
        endo;
        v1[i]=v1[i]./sqrt(T-R-tao[i]+1);
        v2[i]=v2[i]./sqrt(T-R-tao[i]+1);
        vt[i]=vt[i]./sqrt(T-R-tao[i]+1);

    endo;
retp(vt~v1~v2);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*get simulated conditonal density  : simulate latent variables
each step

format:  prob(u1<x_sim(t+tao)<u2|x(t))= sumc
(I(u1<x_sim(t+tao)<u2|x(t)))/S
                                                          *

****************************************************************
*************************************************
* Inputs:       xt      -       time series, The first R
observations that used for estimation
                                                *
*                   tao     -           indicating the different
step ahead con. int. to be examined
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV or MV part
            model_ind -     model indicator
*                   h       -           discretization interval,
for each step,make it 1/h small intervals
                                                *
*                   u_bar   -           confidence interval
            see     -       simulation seed
                                                            *
* Output:   p_sim      -        generalized residual ,
                    *
*                                                   *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)= Con_den(xt,tao,S,N,model_ind,h,u_bar,see);
local
x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,b,mvSim,sv,svSim;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]
        if model_ind==1;
            b=para1[rows(xt)-R+1,.];
```

```
            elseif model_ind==2;
            b=para2[rows(xt)-R+1,.];
            elseif model_ind==3;
            b=para3[rows(xt)-R+1,.];
        endif;

        xtt=xt[rows(xt),.];// the initial value for simulation
        ii=0;
        do while ii<S;ii=ii+1;
            x_sim={}; x_sim2={}; mvSim={}; mv={};
            see=see+3;// change seed to change each path;

            if model_ind==1;
                x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt);
// simulate based on Xt(i), simulat S times;
                p_sim=((x_sim[tao] > u_bar[1,1]).*(x_sim[tao]<
u_bar[2,1]))';

            elseif model_ind==2;

{x_sim2,svSim}=DGP_SV(N,h,b[1],b[2],b[3],b[4],b[5],b[6],see);//s
imulate N times to get mvSim
                for j(1,N,1);

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,svSim[j]);// we need to put 2 starts here, one for Xt, one for
Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                see=see+3;
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==3;

{x_sim2,svSim}=DGP_SVJ(N,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b[
8],b[9],b[10],see);//simulate N times to get mvSim
                for j(1,N,1);

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,svSim[j]);// we need to put 2 starts
here, one for Xt, one for Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==4;

{x_sim2,svSim,mvSim}=DGP_chen(N,h,b[1],b[2],b[3],b[4],b[5],b[6],
b[7],see);//simulate N times to get mvSim
                for j(1,N,1);
                    for jj(1,N,1);

{x_sim,sv,mv}=DGP_chensim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[
```

```
7],see,xtt,svSim[j],mvSim[jj]);// we need to put 2 starts here, 
one for Xt, one for Vt
                         p_sim=p_sim +((x_sim[tao] > 
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==5;

{x_sim2,svSim,mvSim}=DGP_chenj(N,h,b[1],b[2],b[3],b[4],b[5],b[6]
,b[7],b[8],b[9],b[10],b[11],see);//simulate N times to get mvSim

                for j(1,N,1);
                    for jj(1,N,1);

{x_sim,sv,mv}=DGP_chenjsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b
[7],b[8],b[9],b[10],b[11],see,xtt,svSim[j],mvSim[jj]);// we need 
to put 2 starts here, one for Xt, one for Vt
                         p_sim=p_sim +((x_sim[tao] > 
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==6;

{x_sim2,mvSim}=DGP_SM(N,h,b[1],b[2],b[3],b[4],b[5],see);//simula
te N times to get mvSim
                for j(1,N,1);

{x_sim,mv}=DGP_SMsim(tao,h,b[1],b[2],b[3],b[4],b[5],see,xtt,mvSi
m[j]);// we need to put 2 starts here, one for Xt, one for Vt
                     p_sim=p_sim +((x_sim[tao] > 
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;
            endif;

            // calculate the probability of 
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations
              pp_sim=pp_sim +p_sim;

        endo;
pp_sim=pp_sim/S;
retp(pp_sim);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*get simulated conditonal density: take simulated series as 
input
                                   using same parameters

format:  prob(u1<x_sim(t+tao)<u2|x(t))= sumc
```

```
(I(u1<x_sim(t+tao)<u2|x(t)))/S
                                 *

*****************************************************************
*************************************************
* Inputs:       xt      -       time series, The first t
observations that used for estimation
                                                *
*                 tao     -           indicating the different
step ahead con. int. to be examined
*                 S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV or MV part
            model_ind -     model indicator
*                 h       -           discretization interval,
for each step,make it 1/h small intervals
                                                *
*                 u_bar   -           confidence interval
            see     -       simulation seed
            svSim   -       a vector of stochastic volatility
:N*1, act as the start value for SV, SVJ, CHEN,CHENJ simulation.

            mvSim   -       a vector of stochastic mean :N*1,
act as the start value for SM, CHEN, CHENJ simulation.
                                                            *
* Output:   p_sim      -       generalized residual ,
                    *
*                                                   *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)=
Con_den2(xt,tao,S,N,model_ind,h,u_bar,see,svSim,mvSim);
local x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,b,sv;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]
        if model_ind==1;
            b=para1[rows(xt)-R+1,.];
            elseif model_ind==2;
            b=para2[rows(xt)-R+1,.];
            elseif model_ind==3;
            b=para3[rows(xt)-R+1,.];
        endif;
//print "b= " b;
        xtt=xt[rows(xt),.];// the initial value for simulation
//print " the inital value is :" xtt;
        ii=0;
        do while ii<S;ii=ii+1;
            x_sim={}; x_sim2={}; sv={}; mv={};
            see=see+3;// change seed to change each path;

            if model_ind==1;
                x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt);
// simulate based on Xt(i), simulat S times;
```

```
                p_sim=((x_sim[tao] > u_bar[1,1]).*(x_sim[tao]<
u_bar[2,1]))';

            elseif model_ind==2;
                for j(1,N,1);

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,svSim[j]);// we need to put 2 starts here, one for Xt, one for
Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==3;
               for j(1,N,1);

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,svSim[j]);// we need to put 2 starts
here, one for Xt, one for Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==4;
               for j(1,N,1);
                    for jj(1,N,1);

{x_sim,sv,mv}=DGP_chensim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[
7],see,xtt,svSim[j],mvSim[jj]);// we need to put 2 starts here,
one for Xt, one for Vt
                        p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==5;
                for j(1,N,1);
                    for jj(1,N,1);

{x_sim,sv,mv}=DGP_chenjsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b
[7],b[8],b[9],b[10],b[11],see,xtt,svSim[j],mvSim[jj]);// we need
to put 2 starts here, one for Xt, one for Vt
                        p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==6;

{x_sim2,mvSim}=DGP_SM(N,h,b[1],b[2],b[3],b[4],b[5],see);//simula
te N times to get mvSim
```

```
                for j(1,N,1);

{x_sim,mv}=DGP_SMsim(tao,h,b[1],b[2],b[3],b[4],b[5],see,xtt,mvSi
m[j]);// we need to put 2 starts here, one for Xt, one for Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;
            endif;

            // calculate the probability of
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations
              pp_sim=pp_sim +p_sim;

        endo;
pp_sim=pp_sim/S;
retp(pp_sim);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 dont simulate latent variables
proc (1)= Con_den3(xt,tao,S,N,model_ind,h,u_bar,see);

format:  prob(u1<x_sim(t+tao)<u2|x(t))= sumc
(I(u1<x_sim(t+tao)<u2|x(t)))/S
                                *

****************************************************************
*************************************************
* Inputs:       xt      -       time series, The first t
observations that used for estimation
                                                *
*                   tao     -           indicating the different
step ahead con. int. to be examined
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV or MV part
            model_ind -     model indicator
*                   h       -           discretization interval,
for each step,make it 1/h small intervals
                                                *
*                   u_bar   -           confidence interval
            see     -       simulation seed
                                                            *
* Output:   p_sim      -        generalized residual ,
                    *
*                                                   *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)= Con_den3(xt,tao,S,N,model_ind,h,u_bar,see);
local x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,b,sv;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
```

```
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]
        if model_ind==1;
            b=para1[rows(xt)-R+1,.];
            elseif model_ind==2;
            b=para2[rows(xt)-R+1,.];
            elseif model_ind==3;
            b=para3[rows(xt)-R+1,.];
        endif;
//print "b= " b;
        xtt=xt[rows(xt),.];// the initial value for simulation
//print " the inital value is :" xtt;
        ii=0;
        do while ii<S;ii=ii+1;
            x_sim={};sv={}; mv={};
            see=see+3;// change seed to change each path;

            if model_ind==1;
                x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt);
// simulate based on Xt(i), simulat S times;
                p_sim=((x_sim[tao] > u_bar[1,1]).*(x_sim[tao]<
u_bar[2,1]))';

            elseif model_ind==2;

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,b[4]);// we need to put 2 starts here, one for Xt, one for Vt
                    p_sim=((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';

            elseif model_ind==3;

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,b[4]);// we need to put 2 starts here,
one for Xt, one for Vt
                    p_sim=((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
            endif;
            // calculate the probability of
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations

 pp_sim=pp_sim +p_sim;
        endo;
pp_sim=pp_sim/S;
//print "pp_sim=" pp_sim;
retp(pp_sim);
endp;



/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*get simulated conditonal density: bootstrap, using different
estimated parameters
```

```
proc (1)= Con_den4(x,tao,S,N,model_ind,h,u_bar,see,svSim,mvSim);

format:  prob(u1<x_sim(t+tao)<u2|x(t))= sumc
(I(u1<x_sim(t+tao)<u2|x(t)))/S
                                                    *

****************************************************************
**************************************************
* Inputs:       x       -       time series,
                                                                  *
*                   tao     -           indicating the different
step ahead con. int. to be examined
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV or MV part
            model_ind -     model indicator
*                   h       -           discretization interval,
for each step,make it 1/h small intervals
                                                          *
*                   u_bar   -           confidence interval
            see     -       simulation seed
            svSim   -       a vector of stochastic volatility
:N*1, act as the start value for SV, SVJ, CHEN,CHENJ simulation.

            mvSim   -       a vector of stochastic mean :N*1,
act as the start value for SM, CHEN, CHENJ simulation.
                                                              *
* Output:   p_sim       -       generalized residual ,
                    *
*                                                           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)= Con_den4(x,tao,S,N,model_ind,h,u_bar,see,svSim,mvSim);
local x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,b,sv;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]
xt=x;//estimation
b=estimate(xt,model_ind);
//print "b= " b;
        xtt=xt[rows(xt),.];// the initial value for simulation
//print " the inital value is :" xtt;
        ii=0;
        do while ii<S;ii=ii+1;
            x_sim={}; x_sim2={}; sv={}; mv={};
            see=see+3;// change seed to change each path;

            if model_ind==1;
                x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt);
// simulate based on Xt(i), simulat S times;
                p_sim=((x_sim[tao] > u_bar[1,1]).*(x_sim[tao]<
u_bar[2,1]))';

            elseif model_ind==2;
```

```
                for j(1,N,1);

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,svSim[j]);// we need to put 2 starts here, one for Xt, one for
Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==3;
               for j(1,N,1);

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,svSim[j]);// we need to put 2 starts
here, one for Xt, one for Vt
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;

            elseif model_ind==4;
               for j(1,N,1);
                   for jj(1,N,1);

{x_sim,sv,mv}=DGP_chensim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[
7],see,xtt,svSim[j],mvSim[jj]);// we need to put 2 starts here,
one for Xt, one for Vt
                        p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                   endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==5;
                for j(1,N,1);
                   for jj(1,N,1);

{x_sim,sv,mv}=DGP_chenjsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b
[7],b[8],b[9],b[10],b[11],see,xtt,svSim[j],mvSim[jj]);// we need
to put 2 starts here, one for Xt, one for Vt
                       p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                   endfor;
                endfor;
                p_sim=p_sim/(N^2);

            elseif model_ind==6;

{x_sim2,mvSim}=DGP_SM(N,h,b[1],b[2],b[3],b[4],b[5],see);//simula
te N times to get mvSim
                for j(1,N,1);

{x_sim,mv}=DGP_SMsim(tao,h,b[1],b[2],b[3],b[4],b[5],see,xtt,mvSi
m[j]);// we need to put 2 starts here, one for Xt, one for Vt
```

```
                    p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                endfor;
                p_sim=p_sim/N;
            endif;

            // calculate the probability of
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations
              pp_sim=pp_sim +p_sim;

        endo;
pp_sim=pp_sim/S;
retp(pp_sim);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* boot1: Generates the block bootstrap sample
                                  *
*****************************************************************
*******************************
* Inputs:       dat1    -   time series
                                  *
*                   lval    -   block boot length
                                       *
* Output:   xb1     -   bootstrap sample
                                  *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1) = boot1(dat1,lval,see);
local N,num_uns,undraw1,xbl,ib,tt,undraw2;
    N=rows(dat1);
    num_uns=floor(N/lval);

    /* draw uniforms U[0,T-Lval+1] */
    undraw1=round((N-lval)*rndus(num_uns,1,see));
    xbl={};
    ib=1;
    do while ib<=num_uns;
    xbl=xbl|dat1[undraw1[ib]+1:undraw1[ib]+lval,.];
    ib=ib+1;
    endo;
          //handle the left
        see=see+3;// change seed
        tt=N-num_uns*lval;

        if tt>0;
            undraw2=round((N-lval)*rndus(1,1,see));
            xbl=xbl|dat1[undraw2+1:undraw2+tt,.];
        endif;

retp(xbl);
endp;
```

```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS_statBoot2: Calculates The Test Statistics : Simulate latent
ONCE, which need a latent variable series as
            input for the simulation process.
                                                    *
****************************************************************
**************************************************
* Inputs:       xxt      -      time series
            R               The first R observations that used
for estimation
       *
*                   tao     -           a vector indicating the
different step ahead con. int. to be examined
*
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV
  *
*                   model_ind1 -        model specification 1

*                   model_ind2 -        model specification 2
                                *
*                   h       -           discretization interval

*
*                   u_bar   -           confidence interval
            svSim1,mvSim1,svSim2,mvSim2 - simulated series for
latent variables
            option -        =1: using the same parameters
                            =2: estimate new parameters

   *
* Output:
*           vt  -       Test statistics will have rows(tao)
results                                               *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)=
CS_statBoot2(xxt,R,N,tao,S,model_ind1,model_ind2,h,u_bar,svSim1,
mvSim1,svSim2,mvSim2,option);
local
T,tt,xp,p_true,b,f,g,retcode,vt,p_sim1,p_sim2,i,sup_vt,b1,b2,x_s
im1,
x_sim2,see,v1,v2,center1,center2,bb1,bb2,K;
    see=12343;//dont change seed, to use the same random
error,but different start value of X(t)
    T=rows(xxt);

//test
    vt=zeros(rows(tao),1);
    v1=zeros(rows(tao),1);
    v2=zeros(rows(tao),1);
    i=0;
```

```
    do while i <rows(tao);
        i=i+1;
        tt=R;// the start of out-of-sample
        do while tt<=T-tao[i];
            xp=xxt[tt+tao[i],.];// the actual value of X(P), tt
is the in-sample obs
            p_true=(xp .> u_bar[1,1]).*(xp .< u_bar[2,1]);
            xt=xxt[1:tt,.];
            if option==1;

p_sim1=Con_den2(xt,tao[i],S,N,model_ind1,h,u_bar,see,svSim1,mvSi
m1);// prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den2(xt,tao[i],S,N,model_ind2,h,u_bar,see,svSim2,mvSi
m2);// prob(u1<x_sim(t+tao)<u2|x(t))
            elseif option==2;

p_sim1=Con_den3(xt,tao[i],S,N,model_ind1,h,u_bar,see,svSim1,mvSi
m1);// prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den3(xt,tao[i],S,N,model_ind2,h,u_bar,see,svSim2,mvSi
m2);// prob(u1<x_sim(t+tao)<u2|x(t))
            endif;

//get the recenter part, which is the whole sample simulated
against the bootstrap parameters
            if model_ind1==1;
                bb1=para1[tt-R+1,.];
            elseif model_ind1==2;
                bb1=para2[tt-R+1,.];
            elseif model_ind1==3;
                bb1=para3[tt-R+1,.];
            endif;

            if model_ind2==1;
                bb2=para1[tt-R+1,.];
            elseif model_ind2==2;
                bb2=para2[tt-R+1,.];
            elseif model_ind2==3;
                bb2=para3[tt-R+1,.];
            endif;

            center1=0;
            center2=0;
            for k(1,rows(xxt)-tao[i],1);

center1=center1+(Con_denBoot2(xxt[k],tao[i],S,N,model_ind1,h,u_b
ar,see,svSim1,mvSim1,bb1)-p_true)^2/(rows(xxt)-tao[i]);

center2=center2+(Con_denBoot2(xxt[k],tao[i],S,N,model_ind2,h,u_b
ar,see,svSim1,mvSim1,bb2)-p_true)^2/(rows(xxt)-tao[i]);
            endfor;

// calculate the bootstrap statistic
            v1[i]=v1[i]+(p_sim1-p_true).^2-center1;
```

```
            v2[i]=v2[i]+(p_sim2-p_true).^2-center2;
            vt[i]=v1[i]-v2[i];
//print " tt,p_sim1~p_sim2~p_true~v1~v2~vt[i]=="
tt~p_sim1~p_sim2~p_true~v1[i]~v2[i]~vt[i];
            tt=tt+1;
        endo;
        //v1[i]=v1[i]./sqrt(T-R-tao[i]+1);
        //v2[i]=v2[i]./sqrt(T-R-tao[i]+1);
        vt[i]=vt[i]./sqrt(T-R-tao[i]+1);

    endo;
retp(vt);
endp;


/* conditional density for the recenter part in bootstrap */
proc (1)=
Con_denBoot2(x,tao,S,N,model_ind,h,u_bar,see,svSim,mvSim,b);
local x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,sv;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]

            xtt=x;// the initial value for simulation

            ii=0;
            do while ii<S;ii=ii+1;
                x_sim={}; x_sim2={}; sv={}; mv={};
                see=see+3;// change seed to change each path;

                if model_ind==1;

x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt); // simulate
based on Xt(i), simulat S times;
                    p_sim=((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';

                elseif model_ind==2;
                    for j(1,N,1);

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,svSim[j]);// we need to put 2 starts here, one for Xt, one for
Vt
                        p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                    p_sim=p_sim/N;

                elseif model_ind==3;
                   for j(1,N,1);

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,svSim[j]);// we need to put 2 starts
here, one for Xt, one for Vt
                        p_sim=p_sim +((x_sim[tao] >
```

```
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                    p_sim=p_sim/N;

                elseif model_ind==4;
                   for j(1,N,1);
                        for jj(1,N,1);

{x_sim,sv,mv}=DGP_chensim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[
7],see,xtt,svSim[j],mvSim[jj]);// we need to put 2 starts here,
one for Xt, one for Vt
                             p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                        endfor;
                    endfor;
                    p_sim=p_sim/(N^2);

                elseif model_ind==5;
                    for j(1,N,1);
                        for jj(1,N,1);

{x_sim,sv,mv}=DGP_chenjsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b
[7],b[8],b[9],b[10],b[11],see,xtt,svSim[j],mvSim[jj]);// we need
to put 2 starts here, one for Xt, one for Vt
                             p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                        endfor;
                    endfor;
                    p_sim=p_sim/(N^2);

                elseif model_ind==6;

{x_sim2,mvSim}=DGP_SM(N,h,b[1],b[2],b[3],b[4],b[5],see);//simula
te N times to get mvSim
                    for j(1,N,1);

{x_sim,mv}=DGP_SMsim(tao,h,b[1],b[2],b[3],b[4],b[5],see,xtt,mvSi
m[j]);// we need to put 2 starts here, one for Xt, one for Vt
                         p_sim=p_sim +((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
                    endfor;
                    p_sim=p_sim/N;
                endif;

                // calculate the probability of
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations
                  pp_sim=pp_sim +p_sim;

            endo;
    pp_sim=pp_sim/S;

retp(pp_sim);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CS_statBoot3: Calculates The Test Statistics : dont Simulate
latent variables                                            *
****************************************************************
*************************************************
* Inputs:       xxt      -      time series
            R                The first R observations that used
for estimation
       *
*                   tao     -           a vector indicating the
different step ahead con. int. to be examined
*
*                   S       -           number of sample paths
to be simulated
            N       -       number of sample paths to be
simulated for SV
  *
*                   model_ind1 -        model specification 1

*                   model_ind2 -        model specification 2
                                *
*                   h       -           discretization interval

*
*                   u_bar   -           confidence interval
            option -        =1: using the same parameters
                            =2: estimate new parameters

   *
* Output:
*           vt  -       Test statistics will have rows(tao)
results                                              *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (1)=
CS_statBoot3(xxt,R,N,tao,S,model_ind1,model_ind2,h,u_bar,option)
;
local
T,tt,xp,p_true,b,f,g,retcode,vt,p_sim1,p_sim2,i,sup_vt,b1,b2,x_s
im1,
x_sim2,see,v1,v2,center1,center2,bb1,bb2,K;
    see=12343;//dont change seed, to use the same random
error,but different start value of X(t)
    T=rows(xxt);

//test
    vt=zeros(rows(tao),1);
    v1=zeros(rows(tao),1);
    v2=zeros(rows(tao),1);
    i=0;
    do while i <rows(tao);
        i=i+1;
        tt=R;// the start of out-of-sample
        do while tt<=T-tao[i];
            xp=xxt[tt+tao[i],.];// the actual value of X(P), tt
```

```
is the in-sample obs
            p_true=(xp .> u_bar[1,1]).*(xp .< u_bar[2,1]);
            xt=xxt[1:tt,.];
            if option==1;

p_sim1=Con_den3(xt,tao[i],S,N,model_ind1,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den3(xt,tao[i],S,N,model_ind2,h,u_bar,see);//
prob(u1<x_sim(t+tao)<u2|x(t))
            elseif option==2;

p_sim1=Con_den4(xt,tao[i],S,N,model_ind1,h,u_bar,see,svSim1,mvSi
m1);// prob(u1<x_sim(t+tao)<u2|x(t))

p_sim2=Con_den4(xt,tao[i],S,N,model_ind2,h,u_bar,see,svSim2,mvSi
m2);// prob(u1<x_sim(t+tao)<u2|x(t))
            endif;

//get the recenter part, which is the whole sample simulated
against the bootstrap parameters
            if model_ind1==1;
                bb1=para1[tt-R+1,.];
            elseif model_ind1==2;
                bb1=para2[tt-R+1,.];
            elseif model_ind1==3;
                bb1=para3[tt-R+1,.];
            endif;

            if model_ind2==1;
                bb2=para1[tt-R+1,.];
            elseif model_ind2==2;
                bb2=para2[tt-R+1,.];
            elseif model_ind2==3;
                bb2=para3[tt-R+1,.];
            endif;

            center1=0;
            center2=0;
            for k(1,rows(xxt)-tao[i],1);

center1=center1+(Con_denBoot3(xxt[k],tao[i],S,N,model_ind1,h,u_b
ar,see,bb1)-p_true)^2/(rows(xxt)-tao[i]);

center2=center2+(Con_denBoot3(xxt[k],tao[i],S,N,model_ind2,h,u_b
ar,see,bb2)-p_true)^2/(rows(xxt)-tao[i]);
            endfor;

// calculate the bootstrap statistic
            v1[i]=v1[i]+(p_sim1-p_true).^2-center1;
            v2[i]=v2[i]+(p_sim2-p_true).^2-center2;
            vt[i]=v1[i]-v2[i];
//print " tt,p_sim1~p_sim2~p_true~v1~v2~vt[i]=="
tt~p_sim1~p_sim2~p_true~v1[i]~v2[i]~vt[i];
            tt=tt+1;
```

```
        endo;
        //v1[i]=v1[i]./sqrt(T-R-tao[i]+1);
        //v2[i]=v2[i]./sqrt(T-R-tao[i]+1);
        vt[i]=vt[i]./sqrt(T-R-tao[i]+1);

    endo;
retp(vt);
endp;


/* conditional density for the recenter part in bootstrap */
proc (1)= Con_denBoot3(x,tao,S,N,model_ind,h,u_bar,see,b);
local x_sim,x_sim2,p_sim,pp_sim,i,ii,j,jj,mv,xtt,sv;
        p_sim=0; // prob(u1<x_sim<u2) for each simulation, may
be averaged value for SV
        pp_sim=0;// average [ prob(u1<x_sim<u2) ]

            xtt=x;// the initial value for simulation
            ii=0;

        do while ii<S;ii=ii+1;
            x_sim={};sv={}; mv={};
            see=see+3;// change seed to change each path;

            if model_ind==1;
                x_sim=DGP_CIRsim(tao,h,b[1],b[2],b[3],see,xtt);
// simulate based on Xt(i), simulat S times;
                p_sim=((x_sim[tao] > u_bar[1,1]).*(x_sim[tao]<
u_bar[2,1]))';

            elseif model_ind==2;

{x_sim,sv}=DGP_SVsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],see,xtt
,b[4]);// we need to put 2 starts here, one for Xt, one for Vt
                    p_sim=((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';

            elseif model_ind==3;

{x_sim,sv}=DGP_SVJsim(tao,h,b[1],b[2],b[3],b[4],b[5],b[6],b[7],b
[8],b[9],b[10],see,xtt,b[4]);// we need to put 2 starts here,
one for Xt, one for Vt
                    p_sim=((x_sim[tao] >
u_bar[1,1]).*(x_sim[tao]< u_bar[2,1]))';
            endif;
            // calculate the probability of
prob(u1<x_sim(t+tao)<u2|x(t)) for S simulations

         pp_sim=pp_sim +p_sim;
        endo;

    pp_sim=pp_sim/S;

retp(pp_sim);
endp;
```

```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* latentSim: simulation a long serie  for SV using first R obs;.

    *
****************************************************************
*************************************************
* Inputs:       R               In sample obs
            K               The first K observations that used
for estimation
       *
            N       -       number of sample paths to be
simulated for SV
  *
*                   model_ind1 -       model specification 1

*                   model_ind2 -       model specification 2
                                  *
*                   h      -           discretization interval

* Output:
*           svSim1,mvSim1,svSim2,mvSim22: simulated series for
latent variable                                             *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

proc(4)=latentSim(model_ind1,model_ind2,N,h,R,K);
    local x_sim1,svSim1,mvSim1,x_sim2,svSim2,mvSim2,see,b1,b2;
    see=12343;
    x_sim1={};svSim1={};mvSim1={};x_sim2={};svSim2={};mvSim2={};

            if model_ind1==1;
                b1=para1[K-R+1,.];
            elseif model_ind1==2;
                b1=para2[K-R+1,.];
            elseif model_ind1==3;
                b1=para3[K-R+1,.];
            endif;

            if model_ind1==2;

{x_sim1,svsim1}=DGP_SV(N,h,b1[1],b1[2],b1[3],b1[4],b1[5],b1[6],s
ee);//simulate N times to get mvSim
            elseif model_ind1==3;

{x_sim1,svsim1}=DGP_SVJ(N,h,b1[1],b1[2],b1[3],b1[4],b1[5],b1[6],
b1[7],b1[8],b1[9],b1[10],see);//simulate N times to get mvSim
            elseif model_ind1==4;

{x_sim1,svsim1,mvSim1}=DGP_chen(N,h,b1[1],b1[2],b1[3],b1[4],b1[5
],b1[6],b1[7],see);//simulate N times to get mvSim
            elseif model_ind1==5;
```

```
{x_sim1,svsim1,mvSim1}=DGP_chenj(N,h,b1[1],b1[2],b1[3],b1[4],b1[
5],b1[6],b1[7],b1[8],b1[9],b1[10],b1[11],see);//simulate N times
to get mvSim
            elseif model_ind1==6;

{x_sim1,mvSim1}=DGP_SM(N,h,b1[1],b1[2],b1[3],b1[4],b1[5],see);//
simulate N times to get mvSim
            endif;

            if model_ind2==1;
                b2=para1[K-R+1,.];
            elseif model_ind1==2;
                b2=para2[K-R+1,.];
            elseif model_ind1==3;
                b2=para3[K-R+1,.];
            endif;

            if model_ind2==2;

{x_sim2,svSim2}=DGP_SV(N,h,b2[1],b2[2],b2[3],b2[4],b2[5],b2[6],s
ee); //simulate N times to get mvSim
            elseif model_ind2==3;

{x_sim2,svSim2}=DGP_SVJ(N,h,b2[1],b2[2],b2[3],b2[4],b2[5],b2[6],
b2[7],b2[8],b2[9],b2[10],see);//simulate N times to get mvSim
            elseif model_ind2==4;

{x_sim2,svSim2,mvSim2}=DGP_chen(N,h,b2[1],b2[2],b2[3],b2[4],b2[5
],b2[6],b2[7],see);//simulate N times to get mvSim
            elseif model_ind2==5;

{x_sim2,svSim2,mvSim2}=DGP_chenj(N,h,b2[1],b2[2],b2[3],b2[4],b2[
5],b2[6],b2[7],b2[8],b2[9],b2[10],b2[11],see);//simulate N times
to get mvSim
            elseif model_ind2==6;

{x_sim2,mvSim2}=DGP_SM(N,h,b2[1],b2[2],b2[3],b2[4],b2[5],see);//
simulate N times to get mvSim
            endif;

retp(svSim1,mvSim1,svSim2,mvSim2);
endp;
```