Quick Guide to MatTuGames

```
disp('Checking Basic Installation of MatTuGames.');
```

```
Checking Basic Installation of MatTuGames.
```

```
disp('The claims vector is specified by:');
```

```
The claims vector is specified by:
```

```
d=[40 32 11 73.3 54.95 81.1]
```

```
d = 1×6
   40.0000   32.0000   11.0000   73.3000   54.9500   81.1000
```

```
disp('and the estate by:');
```

```
and the estate by:
```

```
E=ceil((3/5)*sum(d))
```

```
E =
176
```

```
disp('The Talmudic distribution is given by:');
```

```
The Talmudic distribution is given by:
```

```
disp('tlm_rl=Talmudic_Rule(E,d)');
```

```
tlm_rl=Talmudic_Rule(E,d)
```

```
tlm_rl=Talmudic_Rule(E,d)
```

```
tlm_rl = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```
disp('The corresponding bankruptcy game is given by:');
```

```
The corresponding bankruptcy game is given by:
```

```
disp('bv=bankruptcy_game(E,d)');
```

```
bv=bankruptcy_game(E,d)
```

```
bv=bankruptcy_game(E,d)
```

```
bv = 1×63
     0     0     0     0     0     0     0     0 ···
```

```
disp('Is the game convex?');
```

```
Is the game convex?
```

```
disp('cvQ=convex_gameQ(bv)');
```

```
cvQ=convex_gameQ(bv)
```

```
cvQ=convex_gameQ(bv)
```

```
cvQ = logical
   1
```

```
disp('Is the core non-empty?');
```

```
Is the core non-empty?
```

```
disp('crQ=coreQ(bv)');
```

```
crQ=coreQ(bv)
```

```
crQ=coreQ(bv)
```

```
crQ = logical
   1
```

```
disp('Is the game monotone?');
```

```
Is the game monotone?
```

```
disp('mQ=monotone_gameQ(bv)');
```

```
mQ=monotone_gameQ(bv)
```

```
mQ=monotone_gameQ(bv)
```

```
mQ = logical
   1
```

```
disp('Is the game zero-monotone?');
```

```
Is the game zero-monotone?
```

```
disp('zmQ=zero_monotonicQ(bv)');
```

```
zmQ=zero_monotonicQ(bv)
```

```
zmQ=zero_monotonicQ(bv)
```

```
zmQ = logical
   1
```

```
disp('Is the game average convex?');
```

```
Is the game average convex?
```

```
disp('acvQ=average_convexQ(bv)');
```

```
acvQ=average_convexQ(bv)
```

```
acvQ=average_convexQ(bv)
```

```
acvQ = logical
   1
```

```
disp('Is the game super additive?');
```

```
Is the game super additive?
```

```
disp('sadQ=super_additiveQ(bv)');
```

```
sadQ=super_additiveQ(bv)
```

```
sadQ=super_additiveQ(bv)
```

```
sadQ = logical
   1
```

```
disp('Is the game semi convex?');
```

Is the game semi convex?

```
disp('scvQ=semi_convexQ(bv)');
```

```
scvQ=semi_convexQ(bv)
```

```
scvQ=semi_convexQ(bv)
```

```
scvQ = logical
   1
```

```
disp('The Harsanyi dividends are given by:');
```

```
The Harsanyi dividends are given by:
```

```
disp('hd=harsanyi_dividends(bv)');
```

```
hd=harsanyi_dividends(bv)
```

```
hd=harsanyi_dividends(bv)
```

```
hd = 1×63
         0         0         0         0         0         0         0         0 ···
```

```
disp('We get the following game from the Harsanyi dividends:');
```

```
We get the following game from the Harsanyi dividends:
```

```
disp('v=getgame(hd)');
```

```
v=getgame(hd)
```

```
v=getgame(hd)
```

```
v = 1×63
         0         0         0         0         0         0         0         0 ···
```

```
tol=10^8*eps;
disp('Coincides this game with the original game bv?');
```

```
Coincides this game with the original game bv?
```

```
geqQ1=all(abs(v-bv)<tol)
```

```
geqQ1 = logical
   1
```

```
disp('The Shapley value of the game is:');
```

```
The Shapley value of the game is:
```

```
disp('sh_v=ShapleyValue(bv)');
```

```
sh_v=ShapleyValue(bv)
```

```
sh_v=ShapleyValue(bv)
```

```
sh_v = 1×6
   23.5175   18.7483    6.4950   44.3008   33.3317   49.6067
```

```
disp('The Tau value of the game is:');
```

```
The Tau value of the game is:
```

```
disp('tau_v=TauValue(bv)');
```

```
tau_v=TauValue(bv)
```

```
tau_v=TauValue(bv)
```

```
tau_v = 1×6
   24.0807   19.2646    6.6222   44.1279   33.0809   48.8237
```

```
disp('The solidarity value of the game is:');
```

```
The solidarity value of the game is:
```

```
disp('sl_vl=SolidarityValue(bv)');
```

```
sl_vl=SolidarityValue(bv)
```

```
sl_vl=SolidarityValue(bv)
```

```
sl_vl = 1×6
   28.0285   27.0298   24.5264   32.5247   30.1953   33.6954
```

```
disp('A pre-kernel element of the game is:');
```

```
A pre-kernel element of the game is:
```

```
disp('prk_v=PreKernel(bv)');
```

```
prk_v=PreKernel(bv)
```

```
prk_v=PreKernel(bv)
```

```
prk_v = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```
disp('A kernel element of the game is:');
```

```
A kernel element of the game is:
```

```
disp('kr_v=Kernel(bv)');
```

```
kr_v=Kernel(bv)
```

```
kr_v=Kernel(bv)
```

```
kr_v = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```
disp('The pre-nucleolus of the game is:');
```

```
The pre-nucleolus of the game is:
```

```
disp('prn_v=PreNucl(bv)');
```

```
prn_v=PreNucl(bv)
```

```
prn_v=PreNucl(bv)
```

```
prn_v = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```
disp('Is the solution found a pre-nucleolus?:');
```

```
Is the solution found a pre-nucleolus?:
```

```
disp('prnQ_v=PrenuclQ(bv,prn_v)');
```

```
prnQ_v=PrenuclQ(bv,prn_v)
```

```
prnQ_v=PrenuclQ(bv,prn_v)
```

```
prnQ_v = logical
   1
```

```
disp('Checking Property II (Kohlberg). Is the induced collection balanced?:');
```

```
Checking Property II (Kohlberg). Is the induced collection balanced?:
```

```
disp('bcQ_v=balancedCollectionQ(bv,prn_v)');
```

```
bcQ_v=balancedCollectionQ(bv,prn_v)
```

```
bcQ=balancedCollectionQ(bv,prn_v)
```

```
bcQ = logical
   1
```

```
disp('The nucleolus of the game is:');
```

```
The nucleolus of the game is:
```

```
disp('nuc_v=nucl(bv)');
```

```
nuc_v=nucl(bv)
```

```
nuc_v=nucl(bv)
```

```
nuc_v = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```
disp('The vector of excesses w.r.t. the pre-kernel is:');
```

```
The vector of excesses w.r.t. the pre-kernel is:
```

```
disp('ex_prk=excess(bv,prk_v)');
```

```
ex_prk=excess(bv,prk_v)
```

```
ex_prk=excess(bv,prk_v)
```

```
ex_prk = 1×63
  -20.0000  -16.0000  -36.0000   -5.5000  -25.5000  -21.5000  -41.5000  -48.3500 ···
```

```
disp('Is the vector prk_v a pre-kernel element?');
```

```
Is the vector prk_v a pre-kernel element?
```

```
disp('prkQ=PrekernelQ(bv,prk_v)');
```

```
prkQ=PrekernelQ(bv,prk_v)
```

```
prkQ=PrekernelQ(bv,prk_v)
```

```
prkQ = logical
   1
```

```
disp('The anti-pre-kernel element of the game is given by:');
```

```
The anti-pre-kernel element of the game is given by:
```

```
disp('aprk=Anti_PreKernel(bv)');
```

```
aprk=Anti_PreKernel(bv)
```

```
aprk=Anti_PreKernel(bv)
```

```
aprk = 1×6
   22.6550   17.9050    7.3050   41.8600   37.1100   49.1650
```

```
disp('The dual game is:');
```

```
The dual game is:
```

```
disp('dv=dual_game(bv)');
```

```
dv=dual_game(bv)
```

```
dv=dual_game(bv)
```

```
dv = 1×63
   40.0000   32.0000   72.0000   11.0000   51.0000   43.0000   83.0000   73.3000 ···
```

```
disp('The greedy bankruptcy game is specified by:');
```

The greedy bankruptcy game is specified by:

```
disp('gv=greedy_bankruptcy(E,d)');
```

```
gv=greedy_bankruptcy(E,d)
```

```
gv=greedy_bankruptcy(E,d)
```

```
gv = 1×63
   40.0000   32.0000   72.0000   11.0000   51.0000   43.0000   83.0000   73.3000 ···
```

```
disp('Is the greedy game equal to the dual?');
```

```
Is the greedy game equal to the dual?
```

```
geqQ2=all(abs(gv-dv)<tol)
```

```
geqQ2 = logical
   1
```

```
disp('A pre-kernel element of the dual is given by:');
```

```
A pre-kernel element of the dual is given by:
```

```
disp('prk_dv=PreKernel(dv)');
```

```
prk_dv=PreKernel(dv)
```

```
prk_dv=PreKernel(dv)
```

```
prk_dv = 1×6
   22.6550   17.9050    7.3050   41.8600   37.1100   49.1650
```

```
disp('An anti-pre-kernel element of the dual is given by:');
```

```
An anti-pre-kernel element of the dual is given by:
```

```
disp('aprk_dv=Anti_PreKernel(dv)');
```

```
aprk_dv=Anti_PreKernel(dv)
```

```
aprk_dv=Anti_PreKernel(dv)
```

```
aprk_dv = 1×6
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500
```

```matlab
disp('Satisfies the pre-kernel consistency?');
```

```
Satisfies the pre-kernel consistency?
```

```matlab
% disp('[RGP RGPC]=Reduced_game_propertyQ(bv,prk_v,PRK)');
echo on
[RGP, RGPC]=Reduced_game_propertyQ(bv,prk_v,'PRK');
echo off;
RGP
```

```
RGP = struct with fields:
    rgpQ: 1
    rgpq: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```matlab
disp('Satisfies the pre-kernel converse consistency?');
```

```
Satisfies the pre-kernel converse consistency?
```

```matlab
echo on;
[CRGP, CRGPC]=Converse_RGP_Q(bv,prk_v,'PRK');
echo off;
CRGP
```

```
CRGP = struct with fields:
    CrgpQ: 1
    crgpQ: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```matlab
disp('Satisfies the pre-kernel element the reconfirmation property?');
```

```
Satisfies the pre-kernel element the reconfirmation property?
```

```matlab
echo on;
[RCP, RCPC]=Reconfirmation_propertyQ(bv,prk_v,'PRK');
echo off;
RCP
```

```
RCP = struct with fields:
    RCPQ: 1
    rcpQ: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
disp('Is the pre-kernel element replicable for some related games?');
```

```
Is the pre-kernel element replicable for some related games?
```

```
disp('RepSol=replicate_prk(bv,prk_v,2,1)');
```

```
RepSol=replicate_prk(bv,prk_v,2,1)
```

```
RepSol=replicate_prk(bv,prk_v,2,1);
RepSol
```

```
RepSol = struct with fields:
        V_PrkQ: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
         V_SPC: [57×63 double]
          SBCQ: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
           SBC: [1×1 struct]
         Mat_W: [16×63 double]
       P_Basis: [57×63 double]
    VarP_Basis: [57×63 double]
           scl: 2
```

```
disp('Select a partition of the player set by:');
```

```
Select a partition of the player set by:
```

```
disp('P={[1 3],[2 4 5],[6]}');
```

```
P={[1 3],[2 4 5],[6]}
```

```
P={[1 3],[2 4 5],[6]}
```

P = 1×3 cell

	1	2	3
1	[1,3]	[2,4,5]	6

```
disp('Transcribe it to its unique integer representation through:');
```

```
Transcribe it to its unique integer representation through:
```

```
disp('pm=clToMatlab(P)');
```

```
pm=clToMatlab(P)
```

```
pm=clToMatlab(P)
```

```
pm = 1×3
     5    26    32
```

```
disp('The Owen value w.r.t. a priori unions pm is:');
```

```
The Owen value w.r.t. a priori unions pm is:
```

```
disp('ow_vl=OwenValue(bv,pm)');
```

```
ow_vl=OwenValue(bv,pm)
```

```
ow_vl=OwenValue(bv,pm)
```

```
ow_vl = 1×6
   21.7083   22.3667    6.4167   46.8500   35.4833   43.1750
```

```
disp('The weighted Owen value w.r.t. a priori unions pm is:');
```

```
The weighted Owen value w.r.t. a priori unions pm is:
```

```
disp('wow_vl=weightedOwen(bv,pm)');
```

```
wow_vl=weightedOwen(bv,pm)
```

```
wow_vl=weightedOwen(v,pm)
```

```
wow_vl = 1×6
   20.4708   29.9500    5.1792   54.4333   43.0667   22.9000
```

```
disp('The coalition solidarity value w.r.t. a priori unions pm is:');
```

```
The coalition solidarity value w.r.t. a priori unions pm is:
```

```
disp('csl_vl=CoalitionSolidarity(bv,pm)');
```

```
csl_vl=CoalitionSolidarity(bv,pm)
```

```
csl_vl=CoalitionSolidarity(bv,pm)
```

```
csl_vl = 1×6
   17.8854   29.8231   10.2396   39.7704   35.1065   43.1750
```

```
disp('Define a communication structure by:');
```

```
Define a communication structure by:
```

```
disp('CS={[1 2],[1 3],[1 4],[2 3],[2 4],[3 4],[4 5],[4 6],[5 6]}');
```

```
CS={[1 2],[1 3],[1 4],[2 3],[2 4],[3 4],[4 5],[4 6],[5 6]}
```

```
CS={[1 2],[1 3],[1 4],[2 3],[2 4],[3 4],[4 5],[4 6],[5 6]}
```

CS = 1×9 cell

	1	2	3	4	5	6	7	8	9
1	[1,2]	[1,3]	[1,4]	[2,3]	[2,4]	[3,4]	[4,5]	[4,6]	[5,6]

```
disp('Transform it to its unique integer representation by:');
```

```
Transform it to its unique integer representation by:
```

```
disp('csm=clToMatlab(CS)');
```

```
csm=clToMatlab(CS)
```

```
csm=clToMatlab(CS)
```

```
csm = 1×9
     3     5     6     9    10    12    24    40    48
```

```
disp(['The Myerson value w.r.t. communication structure csm is specified by:']);
```

```
The Myerson value w.r.t. communication structure csm is specified by:
```

```
disp('my_vl=MyersonValue(bv,csm)');
```

```
my_vl=MyersonValue(bv,csm)
```

```
my_vl=MyersonValue(bv,csm,'cs')
```

```
my_vl = 1×6
   20.7142   17.1158    7.6658   54.7892   31.3200   44.3950
```