Example to test Installation of MatTuGames

Consult also the file `getting_started.m` in the doc-folder.

```
                                    < M A T L A B (R) >
                          Copyright 1984-2013 The MathWorks, Inc.
                            R2013a (8.1.0.604) 64-bit (glnxa64)
                                     February 15, 2013


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

Checking Basic Installation of MatTuGames.
The claims vector is specified by:

d =

   40.0000   32.0000   11.0000   73.3000   54.9500   81.1000

and the estate by:

E =

   176

The Talmudic distribution is given by:
tlm_rl=Talmudic_Rule(E,d)

tlm_rl =

   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

The corresponding bankruptcy game is given by:
bv=bankruptcy_game(E,d)

bv =
```

```
  Columns 1 through 14

       0         0         0         0         0         0         0         0         0         0   28.9500         0    7.9500         0

  Columns 15 through 28

   39.9500         0         0         0   10.6000         0         0         0   21.6000   11.9000   51.9000   43.9000   83.9000   22.9000

  Columns 29 through 42

   62.9000   54.9000   94.9000         0    4.7500         0   36.7500         0   15.7500    7.7500   47.7500   38.0500   78.0500   70.0500

  Columns 43 through 56

  110.0500   49.0500   89.0500   81.0500  121.0500   19.7000   59.7000   51.7000   91.7000   30.7000   70.7000   62.7000  102.7000   93.0000

  Columns 57 through 63

  133.0000  125.0000  165.0000  104.0000  144.0000  136.0000  176.0000

Is the game convex?

cvQ=convex_gameQ(bv)

cvQ =

     1

Is the core non-empty?
crQ=coreQ(bv)
Exiting: The constraints are overly stringent; no feasible starting point found.

crQ =

     1

Is the game monotone?
mQ=monotone_gameQ(bv)

mQ =

     1
```

```
Is the game zero-monotone?
zmQ=zero_monotonicQ(bv)

zmQ =

     1

Is the game average convex?
acvQ=average_convexQ(bv)

acvQ =

     1

Is the game super additive?
sadQ=super_additiveQ(bv)

sadQ =

     1

Is the game semi convex?
scvQ=semi_convexQ(bv)

scvQ =

     1

The Harsanyi dividends are given by:
hd=harsanyi_dividends(bv)

hd =

  Columns 1 through 14

         0         0         0         0         0         0         0         0         0         0   28.9500         0    7.9500         0

  Columns 15 through 28

    3.0500         0         0         0   10.6000         0         0         0   11.0000   11.9000   40.0000   32.0000  -39.5500   11.0000
```

```
  Columns 29 through 42

   -7.9500         0  -14.0500         0    4.7500         0   32.0000         0   11.0000    7.7500   -7.7500   38.0500   35.2500   32.0000

  Columns 43 through 56

  -60.9500   11.0000  -18.9500   -7.7500    4.7000   19.7000   35.2500   32.0000  -42.6000   11.0000  -11.0000   -7.7500   -3.2500   23.3500

  Columns 57 through 63

  -75.2500  -64.0000   71.5500  -22.0000   18.9500    7.7500    6.3000

We get the following game from the Harsanyi dividends:
v=getgame(hd)

v =

  Columns 1 through 14

         0         0         0         0         0         0         0         0         0         0   28.9500         0    7.9500         0

  Columns 15 through 28

   39.9500         0         0         0   10.6000         0         0         0   21.6000   11.9000   51.9000   43.9000   83.9000   22.9000

  Columns 29 through 42

   62.9000   54.9000   94.9000         0    4.7500         0   36.7500         0   15.7500    7.7500   47.7500   38.0500   78.0500   70.0500

  Columns 43 through 56

  110.0500   49.0500   89.0500   81.0500  121.0500   19.7000   59.7000   51.7000   91.7000   30.7000   70.7000   62.7000  102.7000   93.0000

  Columns 57 through 63

  133.0000  125.0000  165.0000  104.0000  144.0000  136.0000  176.0000

Coincides this game with the original game bv?

geqQ1 =

     1
```

```
The Shapley value of the game is:
sh_v=ShapleyValue(bv)

sh_v =

   23.5175   18.7483    6.4950   44.3008   33.3317   49.6067

The Tau value of the game is:
tau_v=TauValue(bv)

tau_v =

   24.0807   19.2646    6.6222   44.1279   33.0809   48.8237

The solidarity value of the game is:
sl_vl=SolidarityValue(bv)

sl_vl =

   28.0285   27.0297   24.5264   32.5247   30.1952   33.6954

A pre-kernel element of the game is:
prk_v=PreKernel(bv)

prk_v =

   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

A kernel element of the game is:
kr_v=Kernel(bv)

kr_v =

   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

The pre-nucleolus of the game is:
prn_v=PreNucl(bv)
Optimization terminated.
Optimization terminated.
Optimization terminated.
Optimization terminated.
```

```
Optimization terminated.
Optimization terminated.
Optimization terminated.

prn_v =

   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

The nucleolus of the game is:
nuc_v=nucl(bv)
Optimization terminated.
Optimization terminated.
Optimization terminated.
Optimization terminated.
Optimization terminated.
Optimization terminated.

nuc_v =

   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

The vector of excesses w.r.t. the pre-kernel is:
ex_prk=excess(bv,prk_v)

ex_prk =

  Columns 1 through 14

  -20.0000  -16.0000  -36.0000   -5.5000  -25.5000  -21.5000  -41.5000  -48.3500  -68.3500  -64.3500  -55.4000  -53.8500  -65.9000  -69.8500

  Columns 15 through 28

  -49.9000  -30.0000  -50.0000  -46.0000  -55.4000  -35.5000  -55.5000  -51.5000  -49.9000  -66.4500  -46.4500  -50.4500  -30.4500  -60.9500

  Columns 29 through 42

  -40.9500  -44.9500  -24.9500  -56.1500  -71.4000  -72.1500  -55.4000  -61.6500  -65.9000  -69.9000  -49.9000  -66.4500  -46.4500  -50.4500

  Columns 43 through 56

  -30.4500  -60.9500  -40.9500  -44.9500  -24.9500  -66.4500  -46.4500  -50.4500  -30.4500  -60.9500  -40.9500  -44.9500  -24.9500  -41.5000
```

```
  Columns 57 through 63

  -21.5000  -25.5000   -5.5000  -36.0000  -16.0000  -20.0000         0

Is the vector prk_v a pre-kernel element?
prkQ=PrekernelQ(bv,prk_v)

prkQ =

     1

The anti-pre-kernel element of the game is given by:
aprk=Anti_PreKernel(bv)

aprk =

   22.6550   17.9050    7.3050   41.8600   37.1100   49.1650

The dual game is:
dv=dual_game(bv)

dv =

  Columns 1 through 14

   40.0000   32.0000   72.0000   11.0000   51.0000   43.0000   83.0000   73.3000  113.3000  105.3000  145.3000   84.3000  124.3000  116.3000

  Columns 15 through 28

  156.3000   54.9500   94.9500   86.9500  126.9500   65.9500  105.9500   97.9500  137.9500  128.2500  168.2500  160.2500  176.0000  139.2500

  Columns 29 through 42

  176.0000  171.2500  176.0000   81.1000  121.1000  113.1000  153.1000   92.1000  132.1000  124.1000  164.1000  154.4000  176.0000  176.0000

  Columns 43 through 56

  176.0000  165.4000  176.0000  176.0000  176.0000  136.0500  176.0000  168.0500  176.0000  147.0500  176.0000  176.0000  176.0000  176.0000

  Columns 57 through 63

  176.0000  176.0000  176.0000  176.0000  176.0000  176.0000  176.0000
```

```
The greedy bankruptcy game is specified by:
gv=greedy_bankruptcy(E,d)

gv =

  Columns 1 through 14

   40.0000   32.0000   72.0000   11.0000   51.0000   43.0000   83.0000   73.3000  113.3000  105.3000  145.3000   84.3000  124.3000  116.3000

  Columns 15 through 28

  156.3000   54.9500   94.9500   86.9500  126.9500   65.9500  105.9500   97.9500  137.9500  128.2500  168.2500  160.2500  176.0000  139.2500

  Columns 29 through 42

  176.0000  171.2500  176.0000   81.1000  121.1000  113.1000  153.1000   92.1000  132.1000  124.1000  164.1000  154.4000  176.0000  176.0000

  Columns 43 through 56

  176.0000  165.4000  176.0000  176.0000  176.0000  136.0500  176.0000  168.0500  176.0000  147.0500  176.0000  176.0000  176.0000  176.0000

  Columns 57 through 63

  176.0000  176.0000  176.0000  176.0000  176.0000  176.0000  176.0000
Is the greedy game equal to the dual?

geqQ2 =

     1

A pre-kernel element of the dual is given by:
prk_dv=PreKernel(dv)

prk_dv =

   22.6550   17.9050    7.3050   41.8600   37.1100   49.1650

An anti-pre-kernel element of the dual is given by:
aprk_dv=Anti_PreKernel(dv)

aprk_dv =
```

```
   20.0000   16.0000    5.5000   48.3500   30.0000   56.1500

Satisfies the pre-kernel consistency?
[RGP RGPC]=Reduced_game_propertyQ(bv,prk_v,'PRK');
echo off;

RGP =

    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]

Satisfies the pre-kernel converse consistency?
[CRGP CRGPC]=Converse_RGP_Q(bv,prk_v,'PRK');
echo off;

CRGP =

    CrgpQ: 1
    crgpQ: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

Satisfies the pre-kernel element the reconfirmation property?
[RCP RCPC]=Reconfirmation_propertyQ(bv,prk_v,'PRK');
echo off;

RCP =

    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]

Is the pre-kernel element replicable for some related games?
RepSol=replicate_prk(bv,prk_v,2,1)

RepSol =

        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: [57x63 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: [1x1 struct]
         Mat_W: [16x63 double]
       P_Basis: [57x63 double]
    VarP_Basis: [57x63 double]
```

```
Select a partition of the player set by:
P={[1 3],[2 4 5],[6]}

P =

    [1x2 double]    [1x3 double]    [6]

Transcribe it to its unique integer representation through:
pm=clToMatlab(P)

pm =

     5    26    32

The Owen value w.r.t. a priori unions pm is:
ow_vl=OwenValue(bv,pm)

ow_vl =

   21.7083   22.3667    6.4167   46.8500   35.4833   43.1750

The weighted Owen value w.r.t. a priori unions pm is:
wow_vl=weightedOwen(bv,pm)

wow_vl =

   20.4708   29.9500    5.1792   54.4333   43.0667   22.9000

The coalition solidarity value w.r.t. a priori unions pm is:
csl_vl=CoalitionSolidarity(bv,pm)

csl_vl =

   17.8854   29.8231   10.2396   39.7704   35.1065   43.1750

Define a communication structure by:
CS={[1 2],[1 3],[1 4],[2 3],[2 4],[3 4],[4 5],[4 6],[5 6]}

CS =

  Columns 1 through 8
```

```
    [1x2 double]    [1x2 double]    [1x2 double]    [1x2 double]    [1x2 double]    [1x2 double]    [1x2 double]    [1x2 double]

  Column 9

    [1x2 double]

Transform it to its unique integer representation by:
csm=clToMatlab(CS)

csm =

     3     5     6     9    10    12    24    40    48

The Myerson value w.r.t. communication structure csm is specified by:
my_vl=MyersonValue(bv,csm)

my_vl =

   20.7142   17.1158    7.6658   54.7892   31.3200   44.3950
```