TUGlab Users Guide

Miguel Ángel Mirás Calvo[1] Estela Sánchez Rodríguez[2]

Universidad de Vigo

March, 2006

[1]Departamento de Matemáticas. Facultad de Ciencias Económicas y Empresariales. Rúa Leonardo da Vinci,s/n. 36310 Vigo. Spain. e-mail: mmiras@uvigo.es

[2]Departamento de Estadística e Investigación Operativa. Facultad de Ciencias Económicas y Empresariales. Rúa Leonardo da Vinci,s/n. 36310 Vigo. Spain. e-mail: esanchez@uvigo.es

Contents

Introduction

The package TUGlab (Transferable Utility Games laboratory) is a Matlab program that could serve as a helpful complement to the books and other materials used in introductory courses on cooperative game theory. Its main goal is to emphasize the geometrical aspects of cooperative game theory. TUGlab offers to both the instructor and the student a tool to compute and visualize basic concepts for any given 3 or 4 persons TU games. It allows the user to experiment at will with games without worrying about the mathematical complexity of the computations. That is the power of this platform: its direct and flexible way of going to the heart of the concepts overcoming the mathematical complexity.

The TUGlab platform works on any implementation of the later releases of the Matlab product: Matlab 6 and Matlab 7 on Unix, PC or Macintosh. It is a collection of 56 files including:

1. The main scripts (29 files) defining the procedures concerning game theory concepts.
2. Auxiliary scripts (27 files) necessary for the computations but not directly related to game theory.
3. Data files (2 files with extension .mat).

Both the main scripts and the auxiliary scripts can be run directly from the Matlab Command Window.

The main scripts are:

1. **additivegame** Checks if a TU game is additive.
2. **admissiblegame** Checks if a TU game is compromise admissible.
3. **balancedgame** Checks if a TU game is balanced.
4. **belongtocore** Checks if a given point belongs to the core of a TU game.
5. **convexgame** Checks if a TU game is convex.
6. **corecenter** Computes the corecenter of a TU game.
7. **corecoverset** Draws the core-cover of a 4-person compromise admissible TU game.

8. **corecoververtices** Computes the vertices of the core-cover of a 4-person compromise admissible TU game.
9. **coreset** Draws the core of a balanced TU game.
10. **corevertices** Computes the vertices of the core of a TU game.
11. **dualgame** Returns the dual game of a TU game.
12. **essentialgame** Checks if a TU game is essential.
13. **excesses** Computes the excesses of an allocation.
14. **harsanyidividends** Computes the Harsanyi dividends of a TU game.
15. **harsanyiset** Draws the Harsanyi set of a TU game.
16. **imputationset** Draws the imputation set of an essential non-degenerate TU game.
17. **imputationvertices** Computes the vertices of the imputation set of a TU game.
18. **MLExtension** Returns the multi-linear extension of a TU game.
19. **monotonicgame** Checks if a TU game is monotonic.
20. **normalizedgame** Provides both the 0 and 0-1 normalizations of a TU game.
21. **nucleolus** Returns the nucleolus of a TU game.
22. **Shapley** Computes the Shapley value and the marginal worth vectors of a TU game.
23. **superadditivegame** Checks if a TU game is superadditive.
24. **tauvalue** Computes the tau-value of a TU game.
25. **totalbalancedgame** Checks if a TU game is totally balanced.
26. **utopiapayoffs** Returns the utopia payoffs of a TU game.
27. **weberset** Draws the Weber set of an essential non-degenerate TU game.
28. **webervertices** Computes the vertices of the Weber set.
29. **zeromonotonicgame** Checks if a TU game is 0-monotonic.

The characteristic function of the game must be introduced as a vector A=[v(1) v(2) v(3) v(12) v(13) v(23) v(123)], for 3 persons games, or A=[v(1) v(2) v(3) v(4) v(12) v(13) v(14) v(23) v(24) v(34) v(123) v(124) v(134) v(234) v(1234)], for 4 persons games. So, for example, the next commands

```
A=[0 0 0 100 200 300 400];[control,info]=convexgame(A)
```

produce the outcome

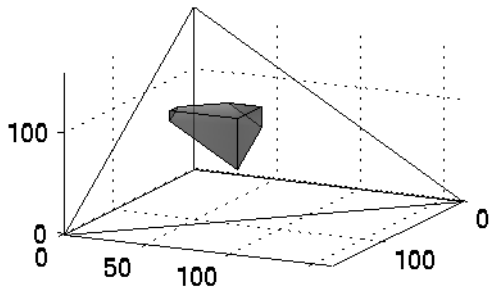
```
control = 0
info = v{123}-v{23}<v{13}-v{3}
```

which tells us that the game given by A is not convex because the inequality $v(123) - v(23) < v(13) - v(3)$ holds.

To draw, for example, the core of a game A it is optimal to plot the imputation set first and then superimpose the core. The following commands

```
A=[0,0,0,0,10,40,30,60,10,20,90,90,120,130,160];
clf
imputationset(A)
hold on,axis(axis)
coreset(A)
```

produce the picture



The 27 auxiliary scripts are:

CCimputation3.m	centroidgame3.m	checkbounds.m
checksegment.m	convexhullextremes.m	corecoverinfo.m
coreinfo.m	efficiency.m	facetsorder.m
gramschmidt.m	harsanyisetinfo.m	heronformula.m
hyperplane.m	imputation3plot.m	imputationset3white.m
linprog.m	lipsol.m	nucleolusaux.m
nucleolusimplex.m	nucleolusinfo.m	poligonorder.m
precision.m	repeatedrows.m	weber4aux.m
weberinfo.m	weberinfoextra.m	weberverticesextra.m

Finally, there are two data files: intersecciones.mat and interseccionesCC.mat.
Please, report any bugs or suggestions about TUGlab to:

mmiras@uvigo.es or esanchez@uvigo.es

All comments would be welcome.

Chapter 1

The main commands

Let us briefly described the 29 main commands defined in the TUGlab platform. For each command we include:

1. The syntax.
2. The Matlab help information.
3. The input variables.
4. The output variables.
5. Some comments regarding the TU game concepts associated with the command and any information that could be of interest to understand how the command operates.
6. An example.
7. A list of related TUGlab commands.

Throughout this manual we use the following notation: A cooperative game with transferable utility, or TU-game, is described by a pair (N, v) where

1. $N = \{1, \ldots, n\}$ is the set of agents.
2. $v\colon 2^N \to \mathbb{R}$ is the characteristic function assigning to each coalition $S \subset N$ a value $v(S)$ representing the benefits from cooperation.

We will always take $v(\emptyset) = 0$. The coalition N is called the grand coalition.

In particular, for 3 and 4 players games, there are 7 and 15 nonempty coalitions respectively, so the corresponding characteristic functions can be identified with vectors in $\mathbb{R}^7$ and $\mathbb{R}^{15}$.

1.1 ADDITIVEGAME

Syntax: [S,info]=additivegame(A)

ADDITIVEGAME Checks if a TU game is additive.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

S=additivegame(A) If the TU game given by vector A is additive S is 1, otherwise S is 0.

[S,info]=additivegame(A) If the game is not additive, info provides an instance where additivity breaks down.
COMMENTS

A game is additive if $v(S \cup T) = v(S) + v(T)$, for all disjoint coalitions $S, T \subset N$.
EXAMPLE

```
>> A=[0 0 0 2 2 2 4];
>> [aditivo,info]=additivegame(A)
aditivo = 0
info =
v13 <> v1+v3
```

See also SUPERADDITIVEGAME, CONVEXGAME.

1.2 ADMISSIBLEGAME

Syntax: [CA,info]=admissiblegame(A)

ADMISSIBLEGAME Checks if a TU game is compromise admissible.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v234 v1234].
OUTPUT

CA=admissiblegame(A) If the TU game given by vector A is compromise admissible, that is, the core-cover is not empty, then CA is 1, otherwise CA is 0.

[CA,info]=admissiblegame(A) If the game is not compromise admissible then the variable info provides an instance where this property fails.
COMMENTS

A game is said to be compromise admissible if the core-cover is a nonempty set.
EXAMPLES

```
>> A=[0 0 0 2 2 2 4];CA=admissiblegame(A)
CA = 1
>> A=[0 0 0 0 1 2 1 1 1 1 4 3 2 1 -2];
>> [CA,info]=admissiblegame(A)
CA =
 0
info =
 m(1)>M(1)
```

See also UTOPIAPAYOFFS, CORECOVERSET, CORECOVERVERTICES, CORECOVERINFO.

1.3 BALANCEDGAME

Syntax: [B,info]=balancedgame(A)

BALANCEDGAME Checks if a TU game is balanced.
INPUT

For 2 players the characteristic function must be introduced with vector A=[v1 v2 v12]. For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

B=balancedgame(A) If the TU game given by vector A is balanced B is 1, otherwise B is 0. Balancedness is determined by the Bondareva-Shapley conditions.

[B,info]=balancedgame(A) If the game is not balanced, i.e., the core is empty, then info provides an instance of a balanced family that does not satisfy the Bondareva-Shapley conditions.
COMMENTS

A game is balanced if the core is a nonempty set. The Bondareva-Shapley conditions are a characterization of balanced games in terms of balanced families of coalitions.
EXAMPLE

```
>> A=[0 0 0 0 1 0 0 0 0 3 3 3 3 3 4];
>> [equilibrado,info]=balancedgame(A)
equilibrado =
 0
info =
 v{1234}<1/2*(v{34}+v{123}+v{124})
```

See also TOTALBALANCEDGAME, CORESET, COREVERTICES, COREINFO, BELONGTOCORE.

1.4 BELONGTOCORE

Syntax: [B,info]=belongtocore(A,v)

BELONGTOCORE Checks if a point belongs to the core of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. Vector v should be a 1×2 vector (the 3rd component of v is computed with the command EFFICIENCY). For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234]. Vector v should be a 1×3 vector (the 4th component of v is is computed with the command EFFICIENCY).
OUTPUT

B=belongtocore(A,v) If point v belongs to the core of the TU game defined by vector A, B is 1, otherwise B is 0.

[B,info]=belongtocore(A,v) If v does not belong to the core, the variable info gives one restriction that defines the core that v does not satisfy.
COMMENTS

The core of a game is the set of those imputations $x \in \mathbb{R}^n$, $x_1 + ... + x_n = v(N)$, such that $x(S) \geq v(S)$ for all coalition S of N, where $x(S) = \sum_{i \in S} x_i$. A vector x belongs to the core if no coalition S has an incentive to split off if x is the proposed reward allocation for N, because the total amount $x(S)$ allocated to S is not smaller than the amount $v(S)$ which they can obtain by forming their own coalition.
EXAMPLE

```
>> A=[0 0 0 2 1 1 6];punto=[0 1];
>> [estable,info]=belongtocore(A,punto)
estable =
 0
info =
 x+y<v{12}
```

See also CORESET, COREVERTICES, COREINFO, EFFICIENCY.

1.5 CONVEXGAME

Syntax: [C,info]=convexgame(A)

CONVEXGAME Checks if a TU game is convex.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

C=convexgame(A) If the TU game given by vector A is convex C is 1, otherwise C is 0.

[C,info]=convexgame(A) If the game is not convex then the variable info provides an instance where convexity fails.
COMMENTS

A game is convex if $v(S\cup T)+v(S\cap T)\geq v(S)+v(T)$, for all coalitions $S,T\subset N$.
EXAMPLE

```
>> A=[0 0 0 0 3 4 4];[convexo,info]=convexgame(A)
convexo =
 0
info =
 v{123}-v{23}<v{13}-v{3}
```

See also SUPERADDITIVEGAME, ADDITIVEGAME.

1.6 CORECENTER

Syntax: centroid=corecenter(A)

CORECENTER The corecenter of a TU balanced game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

centroid=corecenter(A) Computes the corecenter of a TU game given by vector A. The corecenter is defined as the centroid (center of gravity) of the core, so it can only be computed if the game is balanced.
COMMENTS

The corecenter is the expectation of the uniform distribution over the core of the game, i.e., $\mu(v) = E(U)$, being U the uniform distribution over the core $C(v)$.
EXAMPLE

```
>> A=[0 0 0 1 5 5 10];CC=corecenter(A)
CC =
    2.5442    2.5442    4.9116
```

See also CENTROIDGAME3, HERONFORMULA, SHAPLEY, NUCLEOLUS, TAUVALUE, BALANCEDGAME.

1.7 CORECOVERSET

Syntax: corecoverset(A)

CORECOVERSET Draws the core-cover of a 4-person compromise admissible TU game.
INPUT

For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

It displays the core-cover of the 4-person game defined by A. The core-cover will be depicted in yellow. If the game is not compromise admissible then it displays a warning: the core-cover is empty.
COMMENTS

The core-cover consists of all the imputations which are between $m(v)$ and $M(v)$, the minimum rights vector and the utopia payoff vector, in the usual partial order of $\mathbb{R}^n$. For a 3-person game, the core-cover coincides with the core, so CORECOVERSET only works for 4-person games.
EXAMPLE

```
>> A=[0 0 0 0 1 2 1 1 1 1 4 3 2 1 7];
>> imputationset(A),hold on,axis(axis)
>> corecoverset(A)

AD=1
```

See also CORECOVERVERTICES, CORECOVERINFO, ADMISSIBLEGAME, UTOPIAPAYOFFS.

1.8 CORECOVERVERTICES

Syntax: ccpoints=corecoververtices(A)

CORECOVERVERTICES Computes the vertices of the core-cover of a TU game.
INPUT

For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

ccpoints=corecoververtices(A) If A is a compromise admissible game, the core cover is the convex hull of the rows of ccpoints; otherwise ccpoints is empty (ccpoints = []).
COMMENTS

The core-cover consists of all the imputations which are between $m(v)$ and $M(v)$, the minimum rights vector and the utopia payoff vector, in the usual partial order of $\mathbb{R}^n$. For a 3-person game, the core-cover coincides with the core.
EXAMPLE

```
>> A=[0 0 0 0 1 2 1 1 1 1 4 3 2 1 7];
>> vertices=corecoververtices(A)
vertices =
     0     0     4     3
     0     5     0     2
     0     5     2     0
     0     4     0     3
     0     3     4     0
     6     0     0     1
     6     0     1     0
     6     1     0     0
     4     0     0     3
     3     0     4     0
     2     5     0     0
```

See also CORECOVERSET, CORECOVERINFO, ADMISSIBLEGAME, UTOPIAPAYOFFS, REPEATEDROWS.

1.9 CORESET

Syntax: coreset(A)

CORESET Draws the core of a balanced TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

It displays the core of the game defined by A.
COMMENTS

The core consists of all the stable imputations, that is,

$$C(v) = \{x \in I(v) : x(S) = \sum_{i \in S} x_i \geq v(S), \forall S \subset N\}.$$

The core allocations provide the agents with an incentive to maintain the grand coalition. The core can be empty.
EXAMPLE

```
>> A=[0 0 0 0 1 2 1 1 1 1 4 3 2 1 7];
>> imputationset(A),hold on,axis(axis)
>> coreset(A)
```

See also COREVERTICES, COREINFO, BALANCEDGAME, TOTALBALANCEDGAME, BELONGTOCORE.

1.10 COREVERTICES

Syntax: [corepoints,CW]=corevertices(A)

COREVERTICES Computes the vertices of the core of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

corepoints = corevertices(A) If game A is balanced, the core is the convex hull of the rows of B; otherwise corepoints is empty

[corepoints,CW] = corevertices(A) CW are the core vertices that belong to the Weber set.
COMMENTS

The core consists of all the stable imputations, that is, the set of all $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ such that $x_1 + \cdots + x_n = v(N)$ and $x(S) \geq v(S)$, $S \subset N$.
EXAMPLE

```
>> A=[0 0 0 0 1 2 1 1 1 1 4 3 2 1 7];
>> [corepoints, CW]=corevertices(A)

corepoints =

         0    1.0000    3.0000    3.0000
         0    1.0000    4.0000    2.0000
         0    2.0000    2.0000    3.0000
         0    2.0000    4.0000    1.0000
    1.0000         0    3.0000    3.0000
    1.0000         0    4.0000    2.0000
    3.0000         0    1.0000    3.0000
    2.0000         0    4.0000    1.0000
    2.0000    2.0000         0    3.0000
    3.0000    1.0000         0    3.0000
    1.0000    2.0000    4.0000         0
    1.0000    5.0000    1.0000         0
    2.0000    1.0000    4.0000         0
         0    4.0000    2.0000    1.0000
    5.0000         0    1.0000    1.0000
    2.0000    4.0000         0    1.0000
    5.0000    1.0000         0    1.0000
    5.0000    1.0000    1.0000         0
    5.5000    0.5000    0.5000    0.5000
```

```
CW =

     0     1     3     3
     0     1     4     2
     0     2     2     3
     0     2     4     1
     1     0     3     3
     1     0     4     2
     3     0     1     3
     2     0     4     1
     2     2     0     3
     3     1     0     3
     1     2     4     0
     1     5     1     0
     2     1     4     0
```

See also CORESET, COREINFO, BALANCEDGAME, TOTALBALANCEDGAME, BELONGTOCORE, REPEATEDROWS.

1.11 DUALGAME

Syntax: v=dualgame(A)

DUALGAME Returns the dual game of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

v=dualgame(A) If A represents the characteristic function of a TU game, v is the characteristic function of its dual game.
COMMENTS

The dual game w of v is define by $w(S) = v(N) - v(N \setminus S)$, for every coalition S of the set of players N.
EXAMPLE

```
>> A=[0 0 0 2 4 4 23];
>> dual=dualgame(A)
dual =
    19    19    21    23    23    23    23
```

See also NORMALIZEDGAME.

1.12 ESSENTIALGAME

Syntax: [E,D]=essentialgame(A)

ESSENTIALGAME Checks if a TU game is essential.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

E=essentialgame(A) If the imputations set of the TU game given by vector A is not empty E is 1, otherwise E is 0.

[E,D]=essentialgame(A) If the TU game given by vector A is degenerate, i.e. $v(N) = \sum_{i \in N} v(\{i\})$, D is 1, otherwise D is 0.
COMMENTS

A game is essential if the imputation set is non-empty, or equivalently if $v(N) \geq v(1) + \cdots + v(n)$. When $v(N) = v(1) + \cdots + v(n)$ the game is called degenerate or inessential.
EXAMPLES

```
>> A=[0 0 0 0 1 1 1 2 3 1 3 3 3 3 8];
>> E=essentialgame(A)
E = 1
>> B=[1 1 1 2 3 4 3];
>> [esencial,degenerado]=essentialgame(B)
esencial = 1
degenerado =  1
```

See also IMPUTATIONSET, IMPUTATIONVERTICES.

1.13 EXCESSES

Syntax: [excesos,coalicion]=excesses(A,x)

EXCESSES Computes the excesses of an allocation.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. Vector x must be 1×3. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234]. Vector x must be 1×4.
OUTPUT

excesos=excesses(A,x) Computes the excess vector corresponding to the allocation x.

[excesos,coalicion]=excesses(A,x) Also provides the coalitions where the excesses are attained.
COMMENTS Given an allocation $x \in \mathbb{R}^n$ the excess of coalition $S \subset N$ with respect to x is defined as $e(S, x) = v(S) - x(S)$.
EXAMPLE

```
>> A=[2 3 4 7 5 9 12]; x=[2.5 5 4.5]
>> [excesos,coalicion]=excesses(A,x)

excesos =  -0.50  -0.50   -0.50  -0.50  -2.00  -2.00


coalicion =  '1'    '3'    '12'    '23'    '2'    '13'
```

See also NUCLEOLUS, NUCLEOLUSAUX, NUCLEOLUSIMPLEX, NUCLEOLUSINFO.

1.14 HARSANYIDIVIDENDS

Syntax: [HD,unanimidad,Aplus,Aminus]=harsanyidividends(A)

HARSANYIDIVIDENDS Computes the Harsanyi dividends of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

HD=harsanyidividends(A) Provides the Harsanyi dividends of game A.

[HD,unanimidad]=harsanyidividends(A) The rows of unanimidad are the unanimity games.

[HD,unanimidad,Aplus,Aminus]=harsanyidividends(A) The game A can be decomposed as A=Aplus-Aminus. Aplus and Aminus are totally positive, convex and disjoint games.
COMMENTS

The Harsanyi dividends are the coordinates of game A in the base formed by the unanimity games. A game is totally positive if all the Harsanyi dividends are non negative.
EXAMPLE

```
>> A=[3 3 3 2 5 9 9];
>> [HD,unanimidad,Aplus,Aminus]=harsanyidividends(A)
HD = 3     3     3    -4    -1     3     2

unanimidad =
     1     0     0     1     1     0     1
     0     1     0     1     0     1     1
     0     0     1     0     1     1     1
     0     0     0     1     0     0     1
     0     0     0     0     1     0     1
     0     0     0     0     0     1     1
     0     0     0     0     0     0     1

Aplus = 3     3     3     6     6     9    14

Aminus = 0     0     0     4     1     0     5
```

See also HARSANYISET, HARSANYISETINFO.

1.15 HARSANYISET

Syntax: harsanyiset(A)

HARSANYISET Draws the Harsanyi set of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

It displays the Harsanyi set of the game defined by A.
COMMENTS

The Harsanyi set of a game A, also called the selectope, is the set of payoffs vectors obtained by all possible distributions of the Harsanyi dividends of all coalitions amongst its members. The Harsanyi set coincides with the core of a convex game known as the Harsanyi mingame.
EXAMPLE

Since, in general, the Harsanyi set is not contained in the imputation set, to have a clear representation of both we recommend to run the following script. It just computes the vertices of the mingame imputation set and then draws its sides (in green), in the coordinates system defined by game A, along with the Harsanyi set and the imputation set of game A.

```
>> A=[0 0 0 2 5 5 10];
>> [HD,una,Aplus,Aminus]=harsanyidividends(A);
>> [mingame maxgame]=harsanyisetinfo(A);
>> vertices=imputationvertices(mingame);
>> [M,minvert]=CCimputation3(A,vertices);
>> ventana=[min(minvert) max(minvert)];
>> ejes=[ventana(1) ventana(3) ventana(2) ventana(4)];
>> clf
>> minvert(end+1,:)=minvert(1,:);
>> plot(minvert(:,1)',minvert(:,2)',':.g')
>> axis(ejes),hold on
>> harsanyiset(A)
>> imputationset3white(A)
```

See also HARSANYISETINFO, HARSANYIDIVIDENDS.

1.16 IMPUTATIONSET

Syntax: imputationset(A)

IMPUTATIONSET Draws the imputation set of an essential non-degenerate TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

It displays the imputation set of the game defined by A.
COMMENTS

The imputations set is defined by

$$I(v) = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : x_i \geq v(\{i\}) \forall i \in N, \sum_{i=1}^{n} x_i = v(N)\}.$$

It is the set of all individually rational and efficient allocations.
EXAMPLE

```
>> A=[0 0 0 0 0 0 0 10 10 10 20 20 20 20 30];
>> imputationset(A)
```

See also IMPUTATIONVERTICES, IMPUTATION3PLOT, IMPUTATIONSET3WHITE, CCIMPUTATION3, EFFICIENCY.

1.17 IMPUTATIONVERTICES

Syntax: [vertices,window]=imputationvertices(A)

IMPUTATIONVERTICES Returns the vertices of the imputation set
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

vertices=imputationvertices(A) the extreme points of the imputation set

[vertices, window]=imputationvertices(A) provides the window limits to draw the imputation set in the imputation simplex. window=[xmin xmax ymin ymax] for 3-players window=[xmin xmax ymin ymax zmin zmax] for 4-players
EXAMPLE

```
>> A=[0 0 0 0 0 0 0 10 10 10 20 20 20 20 30];
>> [vertices, ventana]=imputationvertices(A)
vertices =
     0     0     0    30
     0     0    30     0
    30     0     0     0
     0    30     0     0
ventana =
     0    30     0    30     0    30
```

See also IMPUTATIONSET, IMPUTATION3PLOT, IMPUTATIONSET3WHITE, CCIMPUTATION3.

1.18 MLEXTENSION

Syntax: [MLE,MLEpol]=MLExtension(A)

MLEXTENSION Returns the multi-linear extension of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

MLE=MLExtension(A) MLE is a symbolic variable that contains the multi-linear extension of the game defined by A. MLE is the expression in factor terms.

[MLE,MLEpol]=MLExtension(A) MLEpol is the multi-linear extension given as a expanded polynomial.
COMMENTS

The multilinear extension of the game (N, v) is the function $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by

$$f(x_1, \ldots, x_n) = \sum_{S \subset N} \left\{ \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j) \right\} v(S),$$

for $0 \leq x_i \leq 1$, $i = 1, \ldots, n$.

The Shapley value of a game (N, v) can be computed by means of the multilinear extension f. Indeed,

$$Sh_i = \int_0^1 \frac{\partial f}{\partial x_i}(t, \ldots, t)dt, \ i = 1, \ldots, n.$$

EXAMPLE

```
>> A=[0 0 0 3 3 1 12];[MLE,MLEpol]=MLExtension(A)
MLE =
3*x*y*(1-z)+3*x*(1-y)*z+(1-x)*y*z+12*x*y*z
MLEpol =
3*x*y+5*x*y*z+3*x*z+y*z
```

See also SHAPLEY.

1.19 MONOTONICGAME

Syntax: [M,info]=monotonicgame(A)

MONOTONICGAME Checks if a TU game is monotonic.
INPUT

For 2 players the characteristic function must be introduced with vector A=[v1 v2 v12]. For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

M=monotonicgame(A) If the TU game given by vector A is monotonic M is 1, otherwise M is 0.

[M,info]=monotonicgame(A) If the game is not monotonic, info provides an instance where monotonicity breaks down.
COMMENTS

A game is monotonic if $v(S) \leq v(T)$ for all coalitions S, T such that $S \subset T$.
EXAMPLE

```
>> A=[4 0 0 3 3 1 12]; [monotono,info]=monotonicgame(A)
monotono = 0
info =
v{1}>v{12}
```

See also ZEROMONOTONICGAME.

1.20 NORMALIZEDGAME

Syntax: [N,M]=normalizedgame(A)

NORMALIZEDGAME Provides the 0-normalization and the 0-1 normalization of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

N=normalizedgame(A) N is the 0-normalization of the TU game given by vector A. For 3 players N=[w1 w2 w3 w12 w13 w23 w123]. For 4-players N=[w1 w2 w3 w4 w12 w13 w14 w23 w24 w34 w123 w124 w134 w234 w1234].

[N,M]=normalizedgame(A) M is the 0-1 normalization of the TU game given by vector A. For 3 players M=[u1 u2 u3 u12 u13 u23 u123]. For 4-players M=[u1 u2 u3 u4 u12 u13 u14 u23 u24 u34 u123 u124 u134 u234 u1234].
COMMENTS

The 0-normalization of a TU-game (N, v) is the game given by $v_0(S) = v(S) - \sum_{i \in S} v(i)$ for $S \subset N$. The 0-normalization can be viewed as a change in the values of the individual coalitions without changing the scale. The 0-1 normalization of an TU-game (N, v) is the game given by $u(S) = kv(S) + \sum_{i \in S} \alpha_i$ where $k = \frac{1}{v(N) - \sum_{i \in N} v(i)}$ and $\alpha_i = -kv(i)$, $i \in N$. Therefore, the 0-1 normalization exists if and only if $v(N) - \sum_{i \in N} v(i) \neq 0$.

EXAMPLE

```
>> A=[2 1 0 9 2 1 6];
>> [normal0 normal1]=normalizedgame(A)
normal0 =
     0     0     0     6     0     0     3
normal1 =
     0     0     0     2     0     0     1
```

See also ZEROMONOTONICGAME, DUALGAME.

1.21 NUCLEOLUS

Syntax: [nucleolo,prenucleolo]=nucleolus(A)

NUCLEOLUS The nucleolus of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

nucleolo=nucleolus(A) Computes the nucleolus of a TU game given by vector A. The nucleolus exists and is unique if the game is essential.

[nucleolo,prenucleolo]=nucleolus(A) Also computes the prenucleolus of game A.
COMMENTS

Let (N, v) be a TU-game such that $I(v) \neq \emptyset$. Given an allocation x the complaint, or excess, of coalition S is defined by $e(x, S) = v(S) - x(S)$. Next, let $e(x)$ denote the vector of excesses with its elements arranged in decreasing order. The nucleolus is then the imputation $\eta \in I(v)$ such that

$$e(\eta) \leq_{\text{lex}} e(x), \text{ for all } x \in I(v).$$

The nucleolus is the allocation of the imputation set that minimizes the maximal complaint of all coalitions. Whenever the core is nonempty, the nucleolus belongs to the core. The prenucleolus is the efficient allocation that minimizes the maximal complaint of all coalitions. Whenever the prenucleolus satisfies individual rationality, i.e. it is an imputation, coincides with the nucleolus.
WARNING

The nucleolus function computes the nucleolus of a given 3 or 4 persons TU game by solving a finite number of linear programming problems. Due to the accumulation of round off errors, the algorithm that solves the linear problems might diverge. That is why, a number of messages will be issued concerning the convergence of the simplex algorithm. If one of the optimizations was not terminated successfully, the result presented may not be the nucleolus. In that case you are advised to use the command NUCLEOLUSINFO.
EXAMPLES

```
>> A=[0 0 0 0 3 3 6 8 9 8 6 15 16 17 20];
>> [nucleolo,prenucleolo]=nucleolus(A)

 Optimization terminated successfully.
 Optimization terminated successfully.
 Optimization terminated successfully.
 Optimization terminated successfully.

nucleolo =
```

```
         1.50          3.67          4.67         10.17


prenucleolo =

         1.50          3.67          4.67         10.17

>>   A=[2 1 0 9 2 1 6]
>> [nucleolo,prenucleolo]=nucleolus(A)

 Optimization terminated successfully.
 Optimization terminated successfully.
 Optimization terminated successfully.
 Optimization terminated successfully.
 Optimization terminated successfully.

nucleolo =

    3.5000    2.5000         0


prenucleolo =

    4.2500    3.2500   -1.5000
```

See also SHAPLEY, CORECENTER, TAUVALUE, LINPROG, LINSOL, CHECKBOUNDS, NUCLEOLUSINFO, NUCLEOLUSAUX, NUCLEOLUSIMPLEX, PRECISION.

1.22 SHAPLEY

Syntax: [Sh,WP]=Shapley(A)

SHAPLEY The Shapley value of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234]. For 5 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v5 v12 v13 v14 v15 v23 v24 v25 v34 v35 v45 v123 v124 v125 v134 v135 v145 v234 v235 v245 v345 v1234 v1235 v1245 v1345 v2345 v12345].
OUTPUT

Sh=Shapley(A) Computes the Shapley value S of a TU game given by vector A.

[Sh,WP]=Shapley(A) Also provides the marginal contribution vectors.
COMMENTS

Let $\sigma\colon N \to \{1,\ldots,n\}$ be an ordering of the agents and Π_N be the set of all orderings of N. The marginal vector with respect to the order $\sigma \in \Pi_N$ is

$$m_i^\sigma = v(\{j \in N : \sigma(j) \leq \sigma(i)\}) - v(\{j \in N : \sigma(j) < \sigma(i)\}).$$

The Shapley value $\phi(v)$ is the allocation rule that assigns to each agent i his expected marginal contribution, presuming that each of the $n!$ orders occurs equally likely:

$$\phi_i(v) = \frac{1}{n!} \sum_{\sigma \in \Pi_N} m_i^\sigma,\ i \in N.$$

EXAMPLE

```
>> A=[0 0 0 6 5 5 10];
>> [Sh,contribuciones]=Shapley(A)
Sh =
    3.5000    3.5000    3.0000
contribuciones =
     0     6     4
     0     5     5
     6     0     4
     5     0     5
     5     5     0
     5     5     0
```

See also NUCLEOLUS, CORECENTER, TAUVALUE.

1.23 SUPERADDITIVEGAME

Syntax: [S,info]=superadditivegame(A)

SUPERADDITIVEGAME Checks if a TU game is superadditive.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

S=superadditivegame(A) If the TU game given by vector A is superadditive S is 1, otherwise S is 0.

[S,info]=superadditivegame(A) If the game is not superadditive, info provides an instance where superadditivity breaks down.
COMMENTS

A game is superadditive if $v(S \cup T) \geq v(S)+v(T)$, for all disjoint coalitions $S, T \subset N$.
EXAMPLE

```
>> A=[1 2 3 2 4 5 6];
>> [super,info]=superadditivegame(A)
super =  0
info =
(v12-v2)<v1
```

See also ADDITIVEGAME, CONVEXGAME.

1.24 TAUVALUE

Syntax: [tau,alfa]=tauvalue(A)

TAUVALUE The tau-value of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

tau=tauvalue(A) Computes the tau-value of a TU game given by vector A. The tau-value can only be computed if the game is of admissible compromise.

[tau,alfa]=tauvalue(A) alfa is the scalar number such that tau=m+alfa*(M-m), where M and m are the utopia payoffs and the minimum rights vectors respectively.
COMMENTS

The τ-value is defined as the efficient allocation τ, i.e. $\sum_{i \in N} \tau_i = v(N)$, such that $\tau = m + \alpha(M - m)$ for some α, where M and m are the utopia payoffs and the minimum rights vectors respectively. The τ-value can only be computed if the game is of admissible compromise.
EXAMPLE

```
>> A=[0 0 0 9 4 7 11];
>> [tau,alfa]=tauvalue(A)
tau =
    3.3333    6.3333    1.3333
alfa =
    0.6667
```

See also ADMISSIBLEGAME, UTOPIAPAYOFFS, NUCLEOLUS, CORECENTER, SHAPLEY.

1.25 TOTALBALANCEDGAME

Syntax: [TB,info]=totalbalancedgame(A)

TOTALBALANCEDGAME Checks if a TU game is totally balanced.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

TB=totalbalancedgame(A) If the TU game given by vector A is totally balanced TB is 1, otherwise TB is 0. Balancedness is determined by the Bondareva-Shapley conditions.

[TB,info]=totalbalancedgame(A) If the game is not totally balanced, i.e., one subgame has empty core, then info provides an instance of a subgame that is not balanced.
COMMENTS

A game is totally balanced if all its subgames are balanced. A subgame of a TU-game (N, v) is a game (R, v_R) such that $R \subset N$ and $v_R(S) = v(S)$ for all $S \subset R$.
EXAMPLE

```
>> A=[0 0 0 0 1 0 0 0 0 0 0 1 0 0 1];
>> [totalequilibrado,info]=totalbalancedgame(A)
totalequilibrado =
     0
info =
The subgame given by the coalition S={1,2,3}
is not balanced, because v{123}<(v{3}+v{12})
```

See also BALANCEDGAME.

1.26 UTOPIAPAYOFFS

Syntax: [M,m]=utopiapayoffs(A)

UTOPIAPAYOFFS The utopia payoffs of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

M=utopiapayoffs(A) Returns the utopia payoffs M of the players of a TU game given by vector A.

[M,m]=utopiapayoffs(A) m is the minimum rights vector of the TU game given by vector A.
COMMENTS

The utopia payoff for player i is given by $M_i = v(N) - v(N \setminus \{i\})$, i.e., is the marginal contribution of player i in the grand coalition. The minimum right for player i is defined by $m_i = \max_{S:i \in S} \left\{ v(S) - \sum_{j \in S \setminus \{i\}} M_j \right\}$.
EXAMPLE

```
>> A=[0 0 0 9 4 7 11];
>> [M,m]=utopiapayoffs(A)
M = 4    7     2

m = 2     5     0
```

See also ADMISSIBLEGAME, TAUVALUE, CORECOVERSET.

1.27 WEBERSET

Syntax: weberset(A)

WEBERSET Draws the Weber set of an essential non-degenerate TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

It displays the Weber set of the game defined by A.
COMMENTS

The Weber set is the convex hull of all the marginal vectors

$$W(v) = \left\{ \sum_{\sigma \in \Pi_N} \alpha_\sigma m^\sigma : \alpha_\sigma \geq 0, \forall \sigma \in \Pi_N; \sum_{\sigma \in \Pi_N} \alpha_\sigma = 1 \right\}.$$

The core is always a subset of the Weber set: $C(v) \subset W(v)$. $C(v) = W(v)$ if and only if the game (N, v) is convex.
EXAMPLE

```
>> A=[0 0 0 0 7 7 7 7 7 7 12 12 12 12 22];
>> imputationset(A), axis(axis),hold on
>> weberset(A)
```

See also WEBERVERTICES, WEBER4AUX, WEBERINFO, WEBERINFOEXTRA, WEBERVERTICESEXTRA, POLIGONORDER, HYPERPLANE, FACETSORDER, CHECKSEGMENT, GRAMSCHMIDT.

1.28 WEBERVERTICES

Syntax: [Wextremes,Wnoextremes]=webervertices(A)

WEBERVERTICES Provides the vertices of the Weber set of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

Wextremes=webervertices(A) Wextremes is a matrix whose rows are the marginal contribution vectors that are extreme points of the Weber set.

[Wextremes,Wnoextremes]=webervertices(A) Wnoextremes are those marginal contribution vectors that are a convex combination of others and, therefore, are not vertices of the Weber set.
COMMENTS

The Weber set is the convex hull of the marginal worth vectors. Naturally, some marginal worth vectors may not be extreme points of the Weber set.
EXAMPLE

```
>> A=[0 0 0 10 4 4 10];
>> [extremos, noextremos]=webervertices(A)
extremos =
     6     0     4
    10     0     0
     0    10     0
     0     6     4
noextremos =
     6     4     0
     4     6     0
```

See also WEBERSET, WEBER4AUX, WEBERINFO, WEBERINFOEXTRA, WEBERVERTICESEXTRA, POLIGONORDER, HYPERPLANE, FACETSORDER, CHECKSEGMENT, GRAMSCHMIDT, REPEATEDROWS.

1.29 ZEROMONOTONICGAME

Syntax: M=zeromonotonicgame(A)

ZEROMONOTONICGAME Checks if a TU game is 0-monotonic.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

M=zeromonotonicgame(A) If the TU game given by vector A is 0-monotonic M is 1, otherwise M is 0.
COMMENTS

A game is 0-monotonic if its 0-normalization is monotonic or, equivalently, if $v(S) + \sum_{i \in T \setminus S} v(i) \leq v(T)$ for all $S \subset T$.
EXAMPLE

```
>> A=[1 0 0 1 1 1 1]
A =
     1     0     0     1     1     1     1

>> M=zeromonotonicgame(A)
M =  0
```

See also MONOTONICGAME, NORMALIZEDGAME.

Chapter 2

The auxiliary commands

The TUGlab toolbox includes 27 auxiliary files, that is, commands that do not define a procedure directly related to game theory but that provide information and perform computations needed to run the main commands. Some of them have an interest of their own and therefore are explained in this chapter.

For each command we include:

1. The syntax.
2. The Matlab help information.
3. The input variables.
4. The output variables.
5. Some comments regarding the TU game concepts associated with the command and any information that could be of interest to understand how the command operates.
6. An example.
7. A list of related TUGlab commands.

2.1 CCIMPUTATION3

Syntax: [M,q]=CCimputation3(A,p)

CCIMPUTATION3 Coordinates in the imputation set triangle of a 3-persons game.
INPUT

The characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. p is a $m \times 3$ matrix whose rows are the true coordinates of m imputation allocations.
OUTPUT

M=CCimputation3(A,p) is the matrix of the linear application that maps the imputation set of game A, a simplex in $\mathbb{R}^3$, into the triangle T in $\mathbb{R}^2$ defined by the vertices $(0,0)$, $(L,0)$ and $(\frac{L}{2}, \frac{\sqrt{3}}{2}L)$, being $L = V(N) - (v(1) + \cdots + v(n))$.

[M,q]=CCimputation3(A,p) The rows of q (a $m \times 2$ matrix) are the images of the rows of p by M,that is, the ith-row of q represents the coordinates of the ith-imputation vector of p in the triangle T.
EXAMPLE

```
>> A=[0 0 0 10 4 4 10];
>> p=[4.3333    4.3333    1.3333; 5 5 0];
>> [M,q]=CCimputation3(A,p)
M =
         0    1.0000    0.5000
         0         0    0.8660
q =

    5.0000    1.1547
    5.0000         0
```

See also IMPUTATIONSET, IMPUTATIONVERTICES, IMPUTATION3PLOT, IMPUTATIONSET3WHITE.

2.2 CENTROIDGAME3

Syntax: v=centroidgame3(A)

CENTROIDGAME3 Auxiliary game to compute the corecenter of a 3-persons game.
INPUT

The characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123].
OUTPUT

v=centroidgame3(A) If A represents the characteristic function of a 3-players TU game, v is the characteristic function of a convex game whose Shapley value is the core-center of game A.
COMMENTS

The game A must be convex with full dimensional core. Otherwise you should avoid dummy players and work with the reduced game since the core-center satisfies the dummy player property.
EXAMPLE

```
>> A=[0 0 0 1 2 3 7];

>>c = centroidgame3(A)

c =   -1.0286   -0.5714   -0.1714   -1.6000   -1.2000

 -0.7429   7.0000
```

See also CORECENTER, SHAPLEY.

2.3 CHECKBOUNDS

Syntax: [x,lb,ub,msg] = checkbounds(xin,lbin,ubin,nvars)

CHECKBOUNDS Move the initial point within the (valid) bounds.
COMMENTS

This file is part of the Matlab Optimization Toolbox. TUGlab uses it in the computation of the Nucleolus to solve several linear programming optimization routines.
NOTE

Revision: 1.5, Date: 2002/03/12 20:36:16.
Mary Ann Branch 5-1-98.

2.4 CHECKSEGMENT

Syntax: [info,neworder]=checksegment(points,order)

CHECKSEGMENT Checks if points are in the same line.
INPUT

points is a 3×2 or 3×3 matrix whose rows are the points we want to know if are aligned. order is a 1×3 vector indicating some ordering (the reference of these points in some matrix) of the 3 points. For instance, points=[0 1 2;0 2 3;0 4 5], order=[1 7 3]. If vector order is not provided then [1 2 3] is used.
OUTPUT

[info,neworder]=checksegment(points,order) info is 1 if the point are in the same line and 0 otherwise. If they are aligned then neworder puts the middle point of the segment in the second place.
EXAMPLE

```
>> points=[0 1 2;0 4 5; 0 2 3];

>> [a,b]=checksegment(points)

a =  1

b =  1     3     2

>> order=[1 7 3]
order = 1     7     3

>> [a,b]=checksegment(points,order)

a =  1

b =  1     3     7
```

See also CONVEXHULLEXTREMES.

2.5 CONVEXHULLEXTREMES

Syntax:
[extremes, noextremes, refextremes, refnoextremes]=
convexhullextremes(points,initial)

CONVEXHULLEXTREMES The extreme points of the convexhull.
INPUT
points is a $m \times 2$ matrix whose rows are the points that determined the convex hull.
initial is a 1xm matrix with the points reference. By default its value is [1:m].
OUTPUT
extremes=convexhullextremes(points,initial) Provides the extreme points of the convex hull defined by the row vectors of matrix points.
[extremes,noextremes]=convexhullextremes(points,initial) noextremes gives the original points that are not extremes of the convex hull.
[extremes, noextremes, refextremes, refnoextremes] = convexhullextremes(points, initial) refextremes and refnoextremes gives the new references.
COMMENTS
The Matlab command CONVHULL computes the points that belong to the border of the convex hull. CONVEXHULLEXTREMES deletes those points which are in the interior of the polytope facets.
EXAMPLE

```
>> points=[6 0 ; 10 0 ; 0 10 ; 0 6 ; 6 4  ; 4 6 ];

>> initial=[6 5 4 3 2 1];

>> [a,b,c,d]=convexhullextremes(points,initial)
a =
     6     0
    10     0
     0    10
     0     6
b =
     6     4
     4     6
c =
     6     5     4     3
d =
     2     1
```

See also CHECKSEGMENT.

2.6 CORECOVERINFO

Syntax:
[CCextremes,hiperplanos,faces,numpuntos,numplanos,
planosactivos,extremosxplano,MPC]=

corecoverinfo(A)
CORECOVERINFO Information needed to draw the core-cover of a TU game.
INPUT

For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

CCextremes=corecoverinfo(A) the extreme points of the core-cover.

hiperplanos= each hyperplane is a 1x3 or 1x4 vector with the coefficients of the equation of the hyperplane of each face of the core-cover. Therefore, [a b c] means ax+by+c=0, and [a b c d] means ax+by+cz+d=0.

faces = matrix of faces ready to be used with the order PATCH (4 players). For instance if the first row is [1 2 3 3 3 3] that means that the first face of the core-cover is formed by the extreme points that are in the positions 1, 2, and 3 in CCextremes.

numpuntos = number of different extreme points.

numplanos = number of faces of the core-cover.

planosactivos = the planes to which at least 3 extreme points belong.

extremosxplano = matrix of size (numpuntos)X8 such that $a_{ij} = 1$ if point i belongs to plane j or $a_{ij} = 0$ if point i does not belong to plane j.

MPC = maximum number of points in any face.
COMMENTS

For a 3-persons game, the core-cover coincides with the core, so COREINFO can be used instead of CORECOVERINFO.
EXAMPLE

```
>> A=[0 0 0 0 7 7 7 7 7 7 12 12 12 12 22];

>> [a,b,c,d,e,f,g]=corecoverinfo(A)

 a =
     0    10    10     2
     0    10     2    10
     0     2    10    10
    10     0    10     2
    10     0     2    10
    10    10     0     2
```

```
    10    10     2     0
    10     2     0    10
    10     2    10     0
     2     0    10    10
     2    10     0    10
     2    10    10     0
b =
     1     0     0     0
    -1     0     0    10
     0     1     0     0
     0    -1     0    10
     0     0     1     0
     0     0    -1    10
     1     1     1   -12
    -1    -1    -1    22
c =
     1     2     3     3     3     3
     5     8     6     7     9     4
     4     5    10    10    10    10
     2    11     6     7    12     1
     6     8    11    11    11    11
     3    10     4     9    12     1
     3    10     5     8    11     2
     7     9    12    12    12    12
d =
    12
e =
     8
f =
     1     2     3     4     5     6     7     8
g =
     1     0     0     1     0     1     0     0
     1     0     0     1     0     0     1     0
     1     0     0     0     0     1     1     0
     0     1     1     0     0     1     0     0
     0     1     1     0     0     0     1     0
     0     1     0     1     1     0     0     0
     0     1     0     1     0     0     0     1
     0     1     0     0     1     0     1     0
     0     1     0     0     0     1     0     1
     0     0     1     0     0     1     1     0
     0     0     0     1     1     0     1     0
     0     0     0     1     0     1     0     1
```

```
h =        6
```

See also See also CORECOVERSET, CORECOVERVERTICES, ADMISSIBLEGAME, UTOPIAPAYOFFS, REPEATEDROWS.

2.7 COREINFO

Syntax:
[Cextremes,hiperplanos,faces,numpuntos,numplanos,
planosactivos,extremosxplano,MPC]=coreinfo(A)

COREINFO provides the information needed to draw the core of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

Cextremes = the extreme points of the core.

hiperplanos = each hyperplane is a 1x3 or 1x4 vector with the coefficients of the equation of the hyperplane of each face of the core-cover. Therefore, [a b c] means ax+by+c=0, and [a b c d] means ax+by+cz+d=0.

faces = ordered extreme points ready to be use with FILL. matrix of faces ready to be used with the order PATCH (For 4 players). For instance, with 4 players, if the first row is [1 2 3 3 3 3 3 3] that means that the first face of the core is formed by the extreme points that are in the positions 1, 2, and 3 in Cextremes.

numpuntos = number of different extreme points.

numplanos = number of faces of the core.

planosactivos = the planes to which at least 3 extreme points belong.

puntosxplano = matrix of size (numpuntos)X14 such that $a_{ij} = 1$ if point i belongs to plane j or $a_{ij} = 0$ if point i does not belong to plane j.

MPC = maximum number of points in any face.
COMMENTS

The core consists of all the stable imputations, that is, the set of all $x = (x_1, ..., x_p)$ such that $x_1 + ... + x_p = v(N)$ and $x(S) \geq v(S)$, $S \subset N$.
EXAMPLE

```
>> A=[0 0 0 6 5 5 10]

>> [a,b,c,d,e,f,g]=coreinfo(A)

a =
     5     5     0
     5     1     4
     1     5     4
b =
     1     0     0
```

```
    -1     0     5
     0     1     0
     0    -1     5
     1     1    -6
    -1    -1    10
c =
     1     2     3
d =
     3
e =
     3
f =
     2     4     5
g =
     0     1     0     1     0     1
     0     1     0     0     1     0
     0     0     0     1     1     0
h=2
```

See also See also CORESET, COREVERTICES, BALANCEDGAME, TOTALBALANCEDGAME, BELONGTOCORE, REPEATEDROWS.

2.8 EFFICIENCY

Syntax: v=efficiency(b,vn)

EFFICIENCY Adds the last component of a vector to be an efficient allocation.
INPUT

b should be a nx2 or nx3 matrix and vn a scalar. The rows of b represent imputations without the last coordinate and vn the value of the grand coalition.
OUTPUT

v=efficiency(b,vn) v is a matrix obtained by adding to matrix b one column, such that the row vectors of v are efficient allocations.
COMMENTS

An allocation $x = (x_1, \ldots, x_n)$ is efficient if $x_1 + \cdots + x_n = v(N)$, where $v(N)$ is the value of the grand coalition. Therefore, the last coordinate of an imputation can always be omitted, because $x_n = v(N) - \sum_{j=1}^{n-1} x_j$.
EXAMPLE

```
>> b=[1 2; 2 4;3 5];

>> vn=10;

>> v=efficiency(b,vn)

v =

     1     2     7
     2     4     4
     3     5     2
```

See also IMPUTATIONSET, IMPUTATIONVERTICES.

2.9 FACETSORDER

Syntax:
[refextremes,refnoextremes,extremes,noextremes]=
facetsorder(facet,points,ref)

FACETSORDER The extreme points of the convex hull of the points in one facet.
INPUT

facet = a 1×4 vector that defines the facet. If facet=[a b c d] then all the points must belong to the plane $ax + by + cz = d$.

points = a $m \times 3$ matrix whose rows are points that belong to the plane defined by vector facet and whose convex hull extreme points we want to compute.

ref= a 1xm matrix with the points reference. By default its value is [1:m].
OUTPUT

[refextremes refnoextremes extremes noextremes]=facetsorder(facet,points,ref) Provides the extreme points of the convex hull defined by the row vectors of matrix points.

refextremes provides the extreme points reference.

refnoextremes provides the non extremes points reference.

extremes gives the 2D-coordinates of the extreme points in the projected space.

noextremes gives the 2D-coordinates of the non extreme points in the projected space.
COMMENTS

The Matlab CONVHULLN command returns the convex hull of a set of given points. In fact, it returns the indices of the points that comprise the facets of the convex hull, and the facets are taken to be triangles. Some of these triangles could define the same facet of the convex hull polyhedron and, as a consequence, some points may not be vertices. The FACETSORDER command takes a number of 3-D points that belong to one plane, projects that plane in the 2-D euclidean space, computes the convex hull of the projected points and brings that information back to the 3-D plane. The projection is done by creating a orthonormal basis of the 3-D euclidean space formed by a basis of the plane and the normal vector. To build it we make use of the Gram-Schmidt method.
EXAMPLE

```
>> facet=[1 2 3 4];
>> points=[0 1 2/3; 1 0 1; 2 2 -2/3; 1/2 1/2 5/6];
>> [a,b,c,d]=facetsorder(facet,points)
a =
     1     2     3
b =
     4
c =
          0          0
    1.4530    0.0000
```

```
    0.3824    2.5752
d =
    0.7265         0
```

See also CONVEXHULLEXTREMES, GRAMSCHMIDT.

2.10 GRAMSCHMIDT

Syntax: ortonormal=gramschmidt(Base)

GRAMSCHMIDT The 3-D Gram-Schmidt orthonormalization method.
INPUT

Base must be a 3×3 matrix whose rows are a base of the 3-dimensional euclidean space.
OUTPUT

The function returns the orthonormal base of the 3-dimensional euclidean space given by the Gram-Schmidt method.
COMMENTS

Given a base $B = \{a_1, a_2, a_3\}$ the Gram-Schmidt method provides an orthonormal base $O = \{b_1, b_2, b_3\}$ such that the subspaces generated by $\{a_1\}$, $\{a_1, a_2\}$ and $\{a_1, a_2, a_3\}$ coincide with the subspaces generated by $\{b_1\}$, $\{b_1, b_2\}$ and $\{b_1, b_2, b_3\}$, respectively.
EXAMPLE

```
>> base=[0 1 2/3; 1 0 1; 2 2 -2/3]

>> gramschmidt(base)

ans =

         0    0.8321    0.5547
    0.7687   -0.3548    0.5322
    0.6396    0.4264   -0.6396
```

See also FACETSORDER.

2.11 HARSANYISETINFO

Syntax: [mingame,maxgame]=harsanyisetinfo(A)

HARSANYISETINFO Provides the information of the Harsanyi set of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

[mingame,maxgame]=harsanyisetinfo(A) Returns the Harsanyi mingame and maxgame of the game A.
COMMENTS

The Harsanyi set H coincides with the core of the mingame which is always a convex game. The mingame m and the maxgame M are defined by

$$m(S) = \min\{x(S) : x \in H\}$$
$$M(S) = \max\{x(S) : x \in H\},$$

for every coalition $S \subset N$.
EXAMPLE

```
>> A=[0 0 0 2 5 5 10];
>> [m,M]=harsanyisetinfo(A)
m =
    -2    -2    -2     0     3     3    10
M =

     7     7    10    12    12    12    10
```

See also HARSANYIDIVIDENS, HARSANYISET.

2.12 HERONFORMULA

Syntax: [area,baricentro]=heronformula(P)

HERONFORMULA Area and barycenter of a triangle using Heron's formula.
INPUT

P must have 3 rows and, at least, 2 columns.
OUTPUT

area=heronformula(P) Computes the area of the triangle whose vertices are given by the row vectors of P. The area is computed using Heron's formula.

[area,baricentro]=heronformula(P) Also provides the barycenter of the given triangle.
COMMENTS

Heron's formula provides the area of a triangle as a function of the lengths of its sides. If a, b and c are the lengths of the 3 sides of a triangle then the area of the triangle is $Area = \sqrt{s(s-a)(s-b)(s-c)}$, where $s = \frac{1}{2}(a+b+c)$.
EXAMPLE

```
>> P=[2 3; 5 3; 7 4]
P =
     2     3
     5     3
     7     4

>> [area,baricentro]=heronformula(P)
area =
    1.5000
baricentro =

    4.6667    3.3333
```

See also CORECENTER.

2.13 HYPERPLANE

Syntax: hiperplano=hyperplane(points)

HYPERPLANE Equation of the n-hyperplane (n=2,3) passing through n points.
INPUT

points is a 3×2 or 3×3 matrix whose rows are the points that determine the hyperplane.
OUTPUT

hiperplano is a 1×3 or 1×4 vector with the coefficients of the equation of the hyperplane passing through the given points. If hiperplano=[a b c] then the equation of the line is $ax + by = c$. If hiperplano=[a b c d] then the equation of the plane is $ax + by + cz = d$.
EXAMPLE

```
>> points=[2 3 5; 5 3 6; 7 4 1];

>> hyperplane(points)
ans =
     1   -17    -3   -64
```

See also POLIGONORDER.

2.14 IMPUTATION3PLOT

Syntax: imputation3plot(A,points,color)

IMPUTATION3PLOT Plotting in the imputations triangle.
INPUT

The characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123].

points is a nx3 matrix whose rows are the points to be plotted.

color is a string defining the color and mark of the points, ready to be used with the command plot. By default color='*b' (asterisk and blue).

OUTPUT

Plots the row points into the imputation triangle of game A.

COMMENTS

To plot a point (x,y,z,t) use the order plot3.

`plot3(x,y,z,'bo')` Draws a blue circle.

`plot3(x,y,z,'k*')` Draws a black asterisk.

`plot3(x,y,z,'g<')` Draws a green sign <.

EXAMPLE

```
>>A =[ 0  0 0  2 5 5 10];

>> imputationset(A)

>> hold on, axis(axis)

>> punto=[5 1 4];

>> imputation3plot(A,punto,'*b')
```

See also See also IMPUTATIONVERTICES, IMPUTATIONSET3WHITE, CCIMPUTATION3.

2.15 IMPUTATIONSET3WHITE

Syntax: imputationset3white(A)

IMPUTATIONSET3WHITE Draws a transparent imputation set for a 3-players TU game.
INPUT
The characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123].
OUTPUT
Plots the imputation set for a 3-players TU game with no background color.
EXAMPLE

```
>> A=[0 0 0 2 5 5 10];

>> imputationset3white(A)
```

See also IMPUTATIONSET, IMPUTATIONVERTICES, IMPUTATION3PLOT, CCIMPUTATION3, EFFICIENCY.

2.16 LINPROG

Syntax: [x,fval,exitflag,output,lambda]=linprog(f,A,B,Aeq,Beq,lb,ub,x0,options)

LINPROG Solves the linear programming problem:

$$\begin{array}{ll} \text{Min} & f' \cdot x \\ \text{subject to :} & \begin{cases} Ax \leq B \\ Aeqx = Beq \\ lb \leq x \leq ub \end{cases} \end{array}$$

COMMENTS

This file is part of the Matlab Optimization Toolbox. TUGlab uses it in the computation of the Nucleolus to solve several linear programming optimization routines.

NOTE

Revision: 1.23, Date: 2002/03/12 20:36:20.

2.17 LIPSOL

Syntax:
[xsol,fval,lambda,exitflag,output]=
lipsol(f,Aineq,bineq,Aeq,beq,lb,ub,options,defaultopt,computeLambda)

LIPSOL Linear programming interior-point solver.
COMMENTS
This file is part of the Matlab Optimization Toolbox. TUGlab uses it in the computation of the Nucleolus to solve several linear programming optimization routines.
NOTE

Revision: 1.23, Date: 2002/03/12 20:36:20.

2.18 NUCLEOLUSAUX

Syntax:
[CC,bb,CCeq,bbeq,indep]=nucleolusaux(C,b,Ceq,beq,saturadas,e,cifras)

NUCLEOLUSAUX Linear programming problems used in nucleolus computation.
INPUT

C, b = The matrix C and vector b are, respectively, the coefficients of the linear inequality constraints and the corresponding right-hand side vector: $C \cdot x \leq b$.

Ceq, beq = The matrix Ceq and vector beq are, respectively, the coefficients of the linear equality constraints and the corresponding right-hand side vector: $Ceq \cdot x \leq beq$.

saturadas = The reference of the binding inequality constraints.

e = The value of the corresponding optimum vector of excesses component.

cifras = The precision (number of accurate decimal places).

OUTPUT

CC, bb = The matrix CC and vector bb are, respectively, the coefficients of the new linear inequality constraints and the corresponding right-hand side vector: $CC \cdot x \leq b$.

CCeq, bbeq = The matrix Ceq and vector bbeq are, respectively, the coefficients of the new linear equality constraints and the corresponding right-hand side vector: $CCeq \cdot x = bbeq$.

indep = The number of independent equality constraints in the new linear programming problem.

COMMENTS

The nucleolus function computes the nucleolus of a given 3 or 4 persons TU game by solving a finite number of linear programming problems. The NUCLEOLUSAUX function takes the linear programing problem solved in the step n and its solution and provides the linear programming problem that should be solved in step $n + 1$. In addition, it computes the number of independent equality constraints in the new linear programming problem. If this number is greater or equal than the number of players, the nucleolus is the solution of the corresponding linear system of equations and the process reaches the end. This function can only be used in conjunction with NUCLEOLUSIMPLEX and NUCLEOLUSINFO.

EXAMPLE

```
>>C =[
    -1     0     0     0    -1
     0    -1     0     0    -1
     0     0    -1     0    -1
     0     0     0    -1    -1
    -1    -1     0     0    -1
    -1     0    -1     0    -1
```

```
    -1     0     0    -1    -1
     0    -1    -1     0    -1
     0    -1     0    -1    -1
     0     0    -1    -1    -1
    -1    -1    -1     0    -1
    -1    -1     0    -1    -1
    -1     0    -1    -1    -1
     0    -1    -1    -1    -1];
>>b =[0 -2 -2 0 -5 -5 0 -5 -2 -2 -9 -5 -5 -5]';
>>Ceq = [1 1 1 1 0]; beq = 11; saturadas =[4 11];
>>e = -1; cifras = 6;
>>[x,y,z,t,u]=nucleolusaux(C,b,Ceq,beq,saturadas,e,cifras)
x =
    -1     0     0     0    -1
     0    -1     0     0    -1
     0     0    -1     0    -1
    -1    -1     0     0    -1
    -1     0    -1     0    -1
    -1     0     0    -1    -1
     0    -1    -1     0    -1
     0    -1     0    -1    -1
     0     0    -1    -1    -1
    -1    -1     0    -1    -1
    -1     0    -1    -1    -1
     0    -1    -1    -1    -1

y =
     0
    -2
    -2
    -5
    -5
     0
    -5
    -2
    -2
    -5
    -5
    -5

z =
     1     1     1     1     0
     0     0     0    -1     0
```

```
t =
    11
    -1
u =
     2
```

See also NUCLEOLUSINFO, NUCLEOLUSIMPLEX, NUCLEOLUS, PRECISION.

2.19 NUCLEOLUSIMPLEX

Syntax:
[reparto,e1,saturadas,sistema]=nucleolusimplex(A,C,b,Ceq,beq,codigo,cifras)

NUCLEOLUSIMPLEX Linear programming algorithm for computing the nucleolus.
INPUT

A = The objective function.

C, b = The matrix C and vector b are, respectively, the coefficients of the linear inequality constraints and the corresponding right-hand side vector: $C \cdot x \leq b$.

Ceq, beq = The matrix Ceq and vector beq are, respectively, the coefficients of the linear equality constraints and the corresponding right-hand side vector: $Ceq \cdot x = beq$.

codigo = 1 if computing the nucleolus, 2 if computing the prenucleolus.

cifras = The precision (number of accurate decimal places).

OUTPUT

reparto = All the components of the optimal solution, except the last one, which give the allocation.

e1 = The last component of the optimal solution, which gives the excess.

saturadas = The binding inequality constraints at the optimum.

sistema = It is a cell that contains the following information: The objective function, matrices C, b, Ceq and beq, the lower and upper constraints of the variables, the vectors reparto, e1, saturadas and convergencia (1 if the algorithm converge, any other number otherwise).

COMMENTS The nucleolus function computes the nucleolus of a given 3 or 4 persons TU game by solving a finite number of linear programming problems. The NUCLEOLUSIMPLEX function takes the linear programing problem of step n and provides its optimal solution. In addition, it saves the information concerning the convergence of the optimization algorithm used. This function can only be used in conjunction with NUCLEOLUSAUX and NUCLEOLUSINFO.

EXAMPLE

```
>>A =[0  2  2  0  5  5  0  5  2  2  9  5  5  5  11];
>>C =[
    -1     0     0     0    -1
     0    -1     0     0    -1
     0     0    -1     0    -1
     0     0     0    -1    -1
    -1    -1     0     0    -1
    -1     0    -1     0    -1
    -1     0     0    -1    -1
     0    -1    -1     0    -1
```

```
     0    -1     0    -1    -1
     0     0    -1    -1    -1
    -1    -1    -1     0    -1
    -1    -1     0    -1    -1
    -1     0    -1    -1    -1
     0    -1    -1    -1    -1];
>>b =[0 -2 -2 0 -5 -5 0 -5 -2 -2 -9 -5 -5 -5]';
>>Ceq =[1     1     1     1     0];
>>beq = 11; codigo = 1; cifras = 6;
>>[r,s,t,u]=nucleolusimplex(A,C,b,Ceq,beq,codigo,cifras)

r =
    2.4066
    3.7967
    3.7967
    1.0000
s =
    -1
t =
     4
    11
u =
  Columns 1 through 4
[1x5 double]  [14x5 double]  [14x1 double]  [1x5 double]
  Columns 5 through 8
[11]  [1x5 double]  [1x5 double]  [4x1 double]
  Columns 9 through 11
[-1.0000]  [2x1 double]  [1]
```

To access the information stored in the cell u just write:

```
>> f=u{1}, C=u{2}, b=u{3}, Ceq=u{4}, beq=u{5}
>> lb=u{6}, ub=u{7}, allocation=u{8}
>> excess=u{9}, binding=u{10}, convergence=u{11}
```

See also NUCLEOLUSAUX, NUCLEOLUSINFO, NUCLEOLUS, PRECISION.

2.20 NUCLEOLUSINFO

Syntax: [nucleolo,infoN,prenucleolo,infoPN]=nucleolusinfo(A,cifras)

NUCLEOLUSINFO Returns the nucleolus and the prenucleolus with a given accuracy.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234]. The natural number cifras indicates the precision (number of accurate decimal places).
OUTPUT

nucleolo = The nucleolus of the given TU game.

infoN = It is a cell that stores the information concerning all the linear programming problems solved to obtain the nucleolus.

prenucleolo = The prenucleolus of the given TU game.

infoPN = It is a cell that stores the information concerning all the linear programming problems solved to obtain the prenucleolus.
COMMENTS

The nucleolus is the allocation of the imputation set that minimizes the maximal complaint of all coalitions. Whenever the core is nonempty, the nucleolus belongs to the core. The prenucleolus is the efficient allocation that minimizes the maximal complaint of all coalitions. Whenever the prenucleolus satisfies individual rationality, i.e. it is an imputation, coincides with the nucleolus.

The nucleolusinfo function computes the nucleolus and the prenucleolus of a given 3 or 4 persons TU game by solving a finite number of linear programming problems. Due to the accumulation of round off errors, the algorithm that solves the linear problems might diverge. That is why, a number of messages will be issued concerning the convergence of the simplex algorithm. If one of the optimizations was not terminated successfully, the result presented may not be the nucleolus. In that case, you are advised to change the accuracy using different values of the input cifras.
EXAMPLE

```
>>A=[0, 0, 0, 0, 0, 1, 2, 2, 3, 0, 2, 3, 2, 3, 4];
>> cifras=6;
>> [a,b,c,d]=nucleolusinfo(A,cifras)

Optimization terminated successfully.
Optimization terminated successfully.
Optimization terminated successfully.
Optimization terminated successfully.

a =
    0.5000   1.5000   0.5000   1.5000
b =
```

```
    {1x11 cell}    {1x11 cell}
c =
    0.5000   1.5000   0.5000   1.5000
d =
    {1x11 cell}    {1x11 cell}
```

b and d are cell arrays of dimension the number of simplex iterations. For instance, the nucleolus computation was completed after solving two linear programming problems. All the information concerning the first one can be obtained writing:

```
>> C=b{1}{1}, Aineq=b{1}{2}, bineq=b{1}{3}, Aeq=b{1}{4},
>> beq=b{1}{5}, lb=b{1}{6},ub=b{1}{7}, sol=b{1}{8},
>> exceso=b{1}{9}, saturadas=b{1}{10},
>> convergencia=b{1}{11}

C =
     0     0     0     0     1

Aineq =
    -1     0     0     0    -1
     0    -1     0     0    -1
     0     0    -1     0    -1
     0     0     0    -1    -1
    -1    -1     0     0    -1
    -1     0    -1     0    -1
    -1     0     0    -1    -1
     0    -1    -1     0    -1
     0    -1     0    -1    -1
     0     0    -1    -1    -1
    -1    -1    -1     0    -1
    -1    -1     0    -1    -1
    -1     0    -1    -1    -1
     0    -1    -1    -1    -1

bineq =

     0
     0
     0
     0
     0
    -1
    -2
```

```
    -2
    -3
     0
    -2
    -3
    -2
    -3
Aeq =
     1     1     1     1     0

beq =
     4
lb =
     0     0     0     0  -Inf
ub =

   Inf   Inf   Inf   Inf   Inf
sol =
    0.5000
    1.5000
    0.5000
    1.5000
exceso =
  1.4293e-009
saturadas =

     6
     7
     8
     9
convergencia =
     1
```

Therefore, in the first step, the NUCLEOLUSINFO function solves the following linear programming problem:

$$\begin{array}{ll} \text{Min} & C \cdot x \\ \text{subject to:} & \begin{cases} Aineq \cdot x \leq bineq \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases} \end{array}$$

where $x = (x_1, x_2, x_3, x_4, e)$. The first four components of the solution (the allocation) are given by sol. The fifth component of the solution (the excess) is stored in exceso. The

binding constraints are given by saturadas. If the simplex converges then convergencia=1, otherwise convergencia=0.

See also NUCLEOLUSAUX, NUCLEOLUSIMPLEX, NUCLEOLUS, PRECISION.

2.21 POLIGONORDER

Syntax: [CO,PO]=poligonorder(C,P)

POLIGONORDER Orders the rows of C to draw a polygon.
INPUT

C is a nx2 o nx3 matrix whose rows are points that belong to a same hyperplane. P must be a 1xn matrix of integer numbers.
OUTPUT

CO=poligonorder(C) CO is obtained by ordering the rows of C so that it is ready to be used with FILL or PATCH to draw a polygon.

[CO,PO]=poligonorder(C,P) if P is a vector that represents the FACES to where the rows of C belong, PO reorders the faces.
COMMENTS

By default P=[1:n]. If C has repeated rows, these will be erased and P will be the default FACES. A warning message will be issued.
EXAMPLE

```
>> C=[0 1 2/3; -1 3 -1/3;1 0 1; 7 7 -17/3; 2 2 -2/3];
>> P=[5 8 3 2 7];
>> [CO,PO]=poligonorder(C,P)

CO =
   -1.0000    3.0000   -0.3333
    7.0000    7.0000   -5.6667
    2.0000    2.0000   -0.6667
    1.0000         0    1.0000
         0    1.0000    0.6667
PO =

     8     2     7     3     5
```

See also HYPERPLANE.

2.22 PRECISION

Syntax: redondeo=precision(numero,cifras)

PRECISION Special rounding to a number of accurate decimal places.
INPUT

The original value (numero) and the number of accurate decimal places (cifras). By default cifras=5.

OUTPUT

The rounded number.

COMMENTS

To control the round off errors when computing the nucleolus we defined a special rounding function.

EXAMPLE

```
>> format long
>> numero=pi;
>> cifras=6;
>> redondeo=precision(numero,cifras)

redondeo =
   3.14159200000000
```

See also NUCLEOLUSINFO, NUCLEOLUSIMPLEX, NUCLEOLUSAUX, NUCLEOLUS.

2.23 REPEATEDROWS

Syntax: [R,T,iguales]=repeatedrows(X,Y)

REPEATEDROWS Erases repeated rows.
INPUT

X and Y must be matrices with the same number of rows.

OUTPUT

R=repeatedrows(X) deletes the repeated rows from matrix X.

[R,T]=repeatedrows(X,Y) deletes the repeated rows of matrix X and the same rows of matrix Y. Naturally, X and Y must have the same number of rows.

[R,T,iguales]=repeatedrows(X,Y) If matrix X has n rows, iguales is a symmetric matrix of order nxn such that iguales(i,j)=1 if rows i and j of X are equal.

COMMENTS

By default Y=[1:size(X,1)]'.

EXAMPLE

```
>> X=[1 2 1 3; 2 4 5 6; 1 2 1 3; 4 6 2 3; 2 4 5 6];
>> Y=[2 3 4 5; 1 2 3 4; 3 4 5 4; 5 6 1 3; 2 3 4 5];
>> [R,T,iguales]=repeatedrows(X,Y)
R =
     1     2     1     3
     2     4     5     6
     4     6     2     3
T =
     2     3     4     5
     1     2     3     4
     5     6     1     3
iguales =
     1     0     1     0     0
     0     1     0     0     1
     1     0     1     0     0
     0     0     0     1     0
     0     1     0     0     1
```

See also CORESET, CORECOVERSET, WEBERSET.

2.24 WEBER4AUX

Syntax:
[W,Wno,ref,refno,faces,hiperplanos,numpuntos,
numplanos,extremosxplano,MPC]=weber4aux(A)

WEBER4AUX provides the information needed to draw the Weber set of a 4 person TU game.
INPUT

For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

W = the extreme points of the Weber set.

Wno = the marginal worth vectors that are not extreme points of the Weber set.

ref = the position of the extreme points in the full list of all marginal worth vectors following the lexicographic order.

refno = the position of the non-extreme points in the full list of all marginal worth vectors following the lexicographic order.

faces = matrix of faces ready to be used with the order PATCH. For instance if the first row is [5 11 13 13] that means that the first face of the weber set is formed by the extreme points that are in the positions 5, 11, and 13 in W.

hiperplanos = the planes where the faces of the Weber set lie. [a b c d] means ax+by+cz=d.

numpuntos = number of different extreme points.

numplanos = number of faces of the Weber set.

extremosxplano = matrix of size (numpuntos)X(numplanos) such that $a_{ij} = 1$ if point i belongs to plane j or $a_{ij} = 0$ if point i does not belong to plane j.

MPC = maximum number of points in any face.
COMMENTS

The Weber set is the convex hull of the marginal worth vectors. Naturally, some marginal worth vectors may not be extreme points of the Weber set.
EXAMPLE

```
>> A=[0, 0, 0, 0, 0, 1, 2, 2, 3, 0, 2, 3, 2, 3, 4];

>> [a,b,c,d,e,f,g,h,i,j]=weber4aux(A)

a =
     0     0     2     2
     0     0     1     3
     0     2     0     2
     1     0     2     1
```

```
     1     0     0     3
     2     2     0     0
     1     3     0     0
     2     1     1     0
     0     3     1     0
b =
     0     1     1     2
     0     2     1     1
     1     1     0     2
     1     2     0     1
c =
     1     2     5     6     7    10    11    12    13
d =
     3     4     8     9
e =
     5    11    13    13
     5     7    10    11
     1     2     5    13
     2     5     7     7
     6     7    12    12
     6    12    13    13
     1     6    13    13
     1     2     7     6
    10    11    13    12
     7    10    12    12
f =
    1.0000   -1.0000    1.0000   -2.0000
         0         0    1.0000         0
    1.0000         0         0         0
    1.0000    0.5000    1.0000    1.0000
    1.0000   -1.0000         0    1.0000
    1.0000    1.0000    2.0000    5.0000
         0    1.0000    3.0000    6.0000
         0    1.0000         0         0
    1.0000    1.0000    1.0000    4.0000
    1.0000   -0.5000   -0.5000    1.0000
g =
     9
h =
    10
i =
    0    0    1    0    0    0    1    1    0    0
    0    0    1    1    0    0    0    1    0    0
```

```
     1     1     1     1     0     0     0     0     0     0
     0     0     0     0     1     1     1     1     0     0
     0     1     0     1     1     0     0     1     0     1
     0     1     0     0     0     0     0     0     1     1
     1     1     0     0     0     0     0     0     1     0
     0     0     0     0     1     1     0     0     1     1
     1     0     1     0     0     1     1     0     1     0
j =
     4
```

See also WEBERSET, WEBERVERTICES, WEBERINFO, WEBERINFOEXTRA, WEBERVERTICESEXTRA.

2.25 WEBERINFO

Syntax:
[Wextremes,faces,hiperplanos,numpuntos,numplanos,
extremosxplano,MPC]=weberinfo(A)

WEBERINFO provides the information needed to draw the Weber set of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

Wextremes = the extreme points of the Weber set.

faces = ordered extreme points ready to be use with FILL. Matrix of faces ready to be used with the order PATCH (For 4 players). For instance if the first row is [1 2] that means that the first face of the weber set is formed by the extreme points that are in the positions 1 and 2 in Wextremes.

hiperplanos = the planes where the faces of the Weber set lie. [a b c] means ax+by=c and [a b c d] means ax+by+cz=d.

numpuntos = number of different extreme points.

numplanos = number of faces of the Weber set.

extremosxplano = matrix of size (numpuntos)X(numplanos) such that $a_{ij} = 1$ if point i belongs to plane j or $a_{ij} = 0$ if point i does not belong to plane j.

MPC = maximum number of points in any face.
COMMENTS

The Weber set is the convex hull of the marginal worth vectors. Naturally, some marginal worth vectors may not be extreme points of the Weber set.
EXAMPLE

```
A=[0, 0, 0, 10, 4, 4, 10];

>> [a,b,c,d,e,f,g]=weberinfo(A)

a =
     6     0     4
    10     0     0
     0    10     0
     0     6     4
b =
     1     2
     2     3
     3     4
```

```
    4    1
c =
    0    1    0
    1    1   10
    1    0    0
    1    1    6
d =
    4
e =
    4
f =
    1    0    0    1
    1    1    0    0
    0    1    1    0
    0    0    1    1
g =
    2
```

See also WEBERSET, WEBERVERTICES, WEBER4AUX, WEBERINFOEXTRA, WEBERVERTICESEXTRA.

2.26 WEBERINFOEXTRA

Syntax:
[Wextremes,faces,hiperplanos,numpuntos,numplanos,
extremosxplano,MPC,Pfaces]=weberinfoExtra(A)

WEBERINFOEXTRA provides the information needed to draw the Weber set of a TU game.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

Wextremes = the extreme points of the Weber set.

faces = ordered extreme points ready to be use with FILL. Matrix of faces ready to be used with the order PATCH (For 4 players). For instance if the first row is [1 2] that means that the first face of the weber set is formed by the extreme points that are in the positions 1 and 2 in Wextremes.

hiperplanos = the planes where the faces of the Weber set lie. [a b c] means ax+by=c and [a b c d] means ax+by+cz=d.

numpuntos = number of different extreme points.

numplanos = number of faces of the Weber set.

extremosxplano = matrix of size (numpuntos)X(numplanos) such that $a_{ij} = 1$ if point i belongs to plane j or $a_{ij} = 0$ if point i does not belong to plane j.

MPC = maximum number of points in any face.

Pfaces = is a cell array that gives the marginal worth vectors that belong to each face of the Weber set. Pfaces is obtained from faces substituting the references of the extreme points by the permutations that defined them.
COMMENTS

The Weber set is the convex hull of the marginal worth vectors. Naturally, some marginal worth vectors may not be extreme points of the Weber set.
EXAMPLE

```
>> A=[0, 0, 0, 0, 0, 1, 2, 2, 3, 0, 2, 3, 2, 3, 4];
>>[a,b,c,d,e,f,g,h]=weberinfoextra(A)
a =
    0    0    2    2
    0    0    1    3
    0    2    0    2
    1    0    2    1
    1    0    0    3
    2    2    0    0
```

```
     1     3     0     0
     2     1     1     0
     0     3     1     0
b =
     5    11    13    13
     5     7    10    11
     1     2     5    13
     2     5     7     7
     6     7    12    12
     6    12    13    13
     1     6    13    13
     1     2     7     6
    10    11    13    12
     7    10    12    12
c =
    1.0000   -1.0000    1.0000   -2.0000
         0         0    1.0000         0
    1.0000         0         0         0
    1.0000    0.5000    1.0000    1.0000
    1.0000   -1.0000         0    1.0000
    1.0000    1.0000    2.0000    5.0000
         0    1.0000    3.0000    6.0000
         0    1.0000         0         0
    1.0000    1.0000    1.0000    4.0000
    1.0000   -0.5000   -0.5000    1.0000
d =
     9
e =
    10
f =
     0     0     1     0     0     0     1     1     0     0
     0     0     1     1     0     0     0     1     0     0
     1     1     1     1     0     0     0     0     0     0
     0     0     0     0     1     1     1     1     0     0
     0     1     0     1     1     0     0     1     0     1
     0     1     0     0     0     0     0     0     1     1
     1     1     0     0     0     0     0     0     1     0
     0     0     0     0     1     1     0     0     1     1
     1     0     1     0     0     1     1     0     1     0
g =
     4
h =
    {1x2 cell}    {1x3 cell}    {1x1 cell}    {1x1 cell}
```

```
   {1x2 cell}   {1x1 cell}   {1x3 cell}   {1x3 cell}
   {1x3 cell}   {1x3 cell}   {1x2 cell}   {1x1 cell}
   {1x3 cell}   {1x2 cell}   {1x1 cell}   {1x1 cell}
   {1x1 cell}   {1x1 cell}   {1x1 cell}   {1x1 cell}
   {1x1 cell}   {1x1 cell}   {1x1 cell}   {1x1 cell}
   {1x3 cell}   {1x1 cell}   {1x1 cell}   {1x1 cell}
   {1x3 cell}   {1x3 cell}   {1x1 cell}   {1x1 cell}
   {1x3 cell}   {1x3 cell}   {1x1 cell}   {1x1 cell}
   {1x1 cell}   {1x3 cell}   {1x1 cell}   {1x1 cell}
```

Now, if you want to see the information contained in the first row of the cell array h, just write

```
 >> h{1,:}

ans =

   '1432'   '3214'
ans =
   '3421'   '4231'   '4321'
ans =
   '4213'
ans =
   '4213'
```

which means that there is one face of the weber set formed by the marginal worth vectors 1432, 3241 and 4213. Besides, the marginal worth vectors 1432 and 3214 coincide and the marginal worth vectors 3241, 4231 and 4321 are the same.

See also WEBERSET, WEBERVERTICES, WEBER4AUX, WEBERINFO, WEBERVERTICESEXTRA.

2.27 WEBERVERTICESEXTRA

Syntax:
[Wextremes,Wnoextremes,contribuciones,repes,repetidos,
Wpoints,Pextremes,PTextremes]=weberverticesExtra(A)

WEBERVERTICESEXTRA Provides the permutations that define the marginal vectors which are extreme points of the Weber set.
INPUT

For 3 players the characteristic function must be introduced with vector A=[v1 v2 v3 v12 v13 v23 v123]. For 4 players the characteristic function must be introduced with vector A=[v1 v2 v3 v4 v12 v13 v14 v23 v24 v34 v123 v124 v134 v234 v1234].
OUTPUT

Wextremes = is a matrix whose rows are the marginal contribution vectors that are extreme points of the Weber set.

Wnoextremes = are those marginal contribution vectors that are a convex combination of others and, therefore, are not vertices of the Weber set.

contribuciones = All the marginal worth vectors ordered according to the lexicographic order of the players permutations. Some vectors can be repeated.

repes = is a symmetric matrix of order nxn such that repes(i,j)=1 if worth vectors i and j are equal.

repetidos = is a cell array 1xp, where p is the number of different marginal vectors, such that repetidos{j} contains the permutations that give rise to the jth worth vector.

Wpoints = The marginal vectors without repetitions.

Pextremes = The permutations that give rise to the extreme points of the Weber set. It is a 1xp cell array.

PTextremes = The permutations, along with those that produced the same worth vectors, that give rise to the extreme points of the Weber set.
COMMENTS

The Weber set is the convex hull of the marginal worth vectors. Naturally, some marginal worth vectors may not be extreme points of the Weber set.
EXAMPLE

```
>> A=[0, 0, 0, 0, 0, 1, 2, 2, 3, 0, 2, 3, 2, 3, 4];
>> [a b c d e f g h]=weberverticesextra(A)
a =
    0    0    2    2
    0    0    1    3
    0    2    0    2
    1    0    2    1
    1    0    0    3
    2    2    0    0
```

```
     1     3     0     0
     2     1     1     0
     0     3     1     0
b =
     0     1     1     2
     0     2     1     1
     1     1     0     2
     1     2     0     1
c =
     0     0     2     2
     0     0     1     3
     0     1     1     2
     0     2     1     1
     0     1     1     2
     0     2     0     2
     0     0     2     2
     0     0     1     3
     0     0     2     2
     1     0     2     1
     0     0     1     3
     1     0     0     3
     1     1     0     2
     1     2     0     1
     0     2     0     2
     1     2     0     1
     2     2     0     0
     1     3     0     0
     2     1     1     0
     2     2     0     0
     0     3     1     0
     1     3     0     0
     2     2     0     0
     1     3     0     0
d =
  Columns 1 through 12
   1   0   0   0   0   0   1   0   1   0   0   0
   0   1   0   0   0   0   0   1   0   0   1   0
   0   0   1   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   1   0   0   0   0   0   0   0
   0   0   0   0   0   1   0   0   0   0   0   0
   1   0   0   0   0   0   1   0   1   0   0   0
   0   1   0   0   0   0   0   1   0   0   1   0
```

```
    1     0     0     0     0     0     1     0     1     0     0     0
    0     0     0     0     0     0     0     0     0     1     0     0
    0     1     0     0     0     0     0     1     0     0     1     0
    0     0     0     0     0     0     0     0     0     0     0     1
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     1     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0

  Columns 13 through 24

    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     1     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    1     0     0     0     0     0     0     0     0     0     0     0
    0     1     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     1     0     0     0     0     0     0     0     0
    0     0     0     0     1     0     0     1     0     0     1     0
    0     0     0     0     0     1     0     0     0     1     0     1
    0     0     0     0     0     0     1     0     0     0     0     0
    0     0     0     0     1     0     0     1     0     0     1     0
    0     0     0     0     0     0     0     0     1     0     0     0
    0     0     0     0     0     1     0     0     0     1     0     1
    0     0     0     0     1     0     0     1     0     0     1     0
    0     0     0     0     0     1     0     0     0     1     0     1
```

```
e =
  Columns 1 through 5

{1x3 cell} {1x3 cell} {1x2 cell} {1x1 cell} {1x2 cell}

  Columns 6 through 10

{1x1 cell} {1x1 cell} {1x1 cell} {1x2 cell} {1x3 cell}

  Columns 11 through 13

{1x3 cell} {1x1 cell} {1x1 cell}

f =
   0   0   2
   0   0   1
   0   1   1
   0   2   1
   0   2   0
   1   0   2
   1   0   0
   1   1   0
   1   2   0
   2   2   0
   1   3   0
   2   1   1
   0   3   1

g =
  Columns 1 through 7

 '1234'  '1243'  '1432'  '2341'  '2431'  '3412'  '3421'

  Columns 8 through 9

 '4123'  '4213'

h =
  Columns 1 through 5

{1x3 cell}  {1x3 cell}  {1x2 cell}  {1x1 cell}  {1x1 cell}
```

```
  Columns 6 through 9

{1x3 cell}  {1x3 cell}  {1x1 cell}  {1x1 cell}
```

See also WEBERSET, WEBERVERTICES, WEBER4AUX, WEBERINFO, WEBERINFOEXTRA.

Bibliography

Curiel, I., 1997. Cooperative game theory and applications. Kluwer Academic Publishers, Dordrecht. The Netherlands.

Derks, J., Kuipers, J., 2002. On the number of extreme points of the core of a transferable utility game. In: Borm, P., Peters, H. (Eds.), Chapters in Game Theory. Kluwer Academic Publishers, Dordrecht. The Netherlands, pp. 83–97.

Driessen, T., 1988. Cooperative games, solutions and applications. Kluwer Academic Publishers, Wiley.

González-Díaz, J., Sánchez-Rodríguez, E., 2003. From set-valued solutions to single-valued solutions: the centroid and the core-center. Working Paper 03-09. Reports in Statistics and Operations Research, Universidad de Santiago de Compostela.

Ichiishi, T., 1981. Supermodularity: applications to convex games and to the greedy algorithm for LP. Journal of Economic Theory 25, 283–286.

Mathworks, 2005. http://www.mathworks.com.

Owen, G., 1995. Game Theory. Academic Press, San Diego.

Rafels i Pallarola, C., Izquierdo i Aznar, J. M., Marín Solano, J., Martínez de Albéniz Salas, F. J., Núñez Oliva, M., Ybern Carballo, N., 1999. Jocs cooperatius i aplicacions econònomiques. Edicions de la Universitat de Barcelona, Barcelona.

Shapley, L., 1971. Cores of convex games. International Journal of Game Theory 1 (3), 11–26.

Vasil´ev, V., van der Laan, G., 2002. The Harsanyi set for cooperative TU-games. Siberian Advances in Mathematics 12, 97–125.

Weber, R. J., 1988. Probabilistic values for games. In: Roth, A. E. (Ed.), The Shapley value. Essays in honor of L. S. Shapley. Cambridge University Press, Cambridge, pp. 101–119.