`IntLinInc3D` package
User manual

Contents:

1 About the package

Necessary software is MATLAB$^{\circledR}$.

The package implements the boundary intervals method [1].

Author of `IntLinInc3D` and the boundary intervals method is Irene A. Sharaya
(Institute of Computational Technologies SB RAS, Novosibirsk).

The package `IntLinInc3D` is free software. Its source codes are open.

Date of the first release is September 1, 2014.

The latest release is available from http://interval.ict.nsc.ru/Programing
and http://interval.ict.nsc.ru/sharaya .

2   Purpose

The package `IntLinInc3D` is intended to visualize various solution sets for interval and point (i.e., noninterval) systems of relations. These systems and solution sets are listed below.

Interval systems:

1) the set of formal solutions for the interval inclusion

$$\boldsymbol{C}x \subseteq \boldsymbol{d} \tag{1}$$

in Kaucher arithmetic, where
$\boldsymbol{C} = [\underline{C}, \overline{C}] \in \mathbb{KR}^{m \times 3}$ is an interval matrix with given endpoints $\underline{C}$ and $\overline{C}$;
$x \in \mathbb{R}^3$ is a real vector of unknowns;
$\boldsymbol{d} = [\underline{d}, \overline{d}] \in \mathbb{KR}^m$ is an interval vector with given endpoints $\underline{d}$ and $\overline{d}$;
$m \in \mathbb{N}$ is a natural (positive integer) number;
$\mathbb{KR} = \{[\underline{z}, \overline{z}] \mid \underline{z}, \overline{z} \in \mathbb{R}\}$ is the set of Kaucher intervals (in contrast to
    the set of classical intervals $\mathbb{IR} = \{[\underline{z}, \overline{z}] \mid \underline{z}, \overline{z} \in \mathbb{R}, \ \underline{z} \leq \overline{z}\}$,
    the requirement $\underline{z} \leq \overline{z}$ is absent for Kaucher intervals);
$\mathbb{K\overline{R}} = \{[\underline{z}, \overline{z}] \mid \underline{z}, \overline{z} \in \overline{\mathbb{R}}\}$ is the set of Kaucher intervals over the extended
    real axis $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$;
multiplication $\boldsymbol{C}$ by x is standard for Kaucher arithmetic;
the inclusion "$\subseteq$" is defined by inequalities $\underline{C}x \geq \underline{d}$ and $\overline{C}x \leq \overline{d}$, which
    are understood componentwise, $\underline{C}x$ and $\overline{C}x$ are the left and right
    endpoints of the interval vector $\boldsymbol{C}x = [\underline{C}x, \overline{C}x]$ respectively;

2) all possible AE-solution sets for the interval system of equations

$$\boldsymbol{A}x = \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{IR}^{m \times 3}, \ \boldsymbol{b} \in \mathbb{IR}^m, \ m \in \mathbb{N}; \tag{2}$$

3) all possible quantifier solution sets for the interval system of inequalities

$$\boldsymbol{A}x \geq \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{IR}^{m \times 3}, \ \boldsymbol{b} \in \mathbb{IR}^m, \ m \in \mathbb{N}, \tag{3}$$

or

$$\boldsymbol{A}x \leq \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{IR}^{m \times 3}, \ \boldsymbol{b} \in \mathbb{IR}^m, \ m \in \mathbb{N}; \tag{4}$$

4) various quantifier solution sets for the interval mixed system of linear equations and inequalities

$$\boldsymbol{A}x \; \sigma \; \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{IR}^{m \times 3}, \; \boldsymbol{b} \in \mathbb{IR}^m, \; \sigma \in \{=, \geq, \leq\}^m, \; m \in \mathbb{N}; \tag{5}$$

specifically, we mean all those solutions for which quantifier description has AE-order of quantifiers for rows with the relation "=".

Point systems:

1) the solution set for the system

$$Ax + B|x| \geq c, \quad A, B \in \mathbb{R}^{m \times 3}, \; c \in \mathbb{R}^m, \; m \in \mathbb{N}; \tag{6}$$

2) the solution set for the system

$$|Ax - c| \leq B|x| + d, \quad A, B \in \mathbb{R}^{m \times 3}, \; c, d \in \mathbb{R}^m, \; m \in \mathbb{N}; \tag{7}$$

3) the solution set for system of linear equations, inequalities and two-sided inequalities

$$\begin{cases} A_{(1)}x = b_{(1)}, & A_{(1)} \in \mathbb{R}^{m_1 \times 3}, \; b_{(1)} \in \mathbb{R}^{m_1}, & m_1 \in \mathbb{N} \cup \{0\}, \\ b_{(2)} \leq A_{(2)}x, & A_{(2)} \in \mathbb{R}^{m_2 \times 3}, \; b_{(2)} \in \mathbb{R}^{m_2}, & m_2 \in \mathbb{N} \cup \{0\}, \\ A_{(3)}x \leq b_{(3)}, & A_{(3)} \in \mathbb{R}^{m_3 \times 3}, \; b_{(3)} \in \mathbb{R}^{m_3}, & m_2 \in \mathbb{N} \cup \{0\}, \\ b_{(4)} \leq A_{(4)}x \leq b_{(5)}, & A_{(4)} \in \mathbb{R}^{m_4 \times 3}, \; b_{(4)}, b_{(5)} \in \mathbb{R}^{m_4}, & m_4 \in \mathbb{N} \cup \{0\}, \end{cases} \tag{8}$$

with $m_1 + m_2 + m_3 + m_4 > 0$.

In [2], it is shown that each solution set listed above can be represented as the set of formal solutions to the inclusion (1). Therefore, the visualization of this set play a key role in the package, which is reflected in the title `IntLinInc3D`, i. e. Interval Linear Inclusion. The last letters `3D` mean that the dimension of the unknowns is 3 ($x \in \mathbb{R}^3$).

Remark. The package `IntLinInc3D` is aimed at illustrating simple examples (in publications, education, etc.), so it works most correctly when the initial data are integers and lie in the range $[-10^2, 10^2]$.

3   Structure

The main function of the package is `Cxind3D`. It is designed to visualize the set of formal solutions for the inclusion (1).

The functions used in the main one are

```
AddV,                ChooseDrawingBox,  DrawHedrons,
BoundaryIntervals,   ClearRows,         Intervals2Path,
ChangeVariables,     ClearZeroRows,     NonRepeatRows.
```

The package contains auxiliary functions for the problems equivalent to (1). The choice of the auxiliary function depends on which of the systems (2)–(7) is to be processed and, for interval systems, on the solution type. The names of the auxiliary functions reflect this dependency.

The names of the auxiliary functions for the interval systems

system	solution type				
	weak	tolerable	controllable	strong	quantifier
(2) $\boldsymbol{A}x = \boldsymbol{b}$	EqnWeak3D	EqnTol3D	EqnCtl3D	EqnStrong3D	EqnAEss3D
(3) $\boldsymbol{A}x \geq \boldsymbol{b}$	GeqWeak3D	GeqTol3D	GeqCtl3D	GeqStrong3D	GeqQtr3D
(4) $\boldsymbol{A}x \leq \boldsymbol{b}$	LeqWeak3D	LeqTol3D	LeqCtl3D	LeqStrong3D	LeqQtr3D
(5) $\boldsymbol{A}x \, \sigma \, \boldsymbol{b}$	MixWeak3D	MixTol3D	MixCtl3D	MixStrong3D	MixQtr3D

The solution types from the table above, except the quantifier type, are defined in IntLinInc2D User manual. A complete set of definitions is in [2], and it generalizes the terminology from [3, 4]. Note that the auxiliary functions `EqnAEss3D` and `MixQtr3D` are designed only for such quantifier solutions which have AE-order of quantifiers in rows with the relation "=".

For point systems, there are two auxiliary functions: the function `Abs13D` is intended for the system (6) with one absolute value operation, the function `Abs23D` is designed for the system (7) which contains two such operations.

We shall refer to the main and auxiliary functions of the package as *launch functions*. Arguments of the launch functions are described in comments within their bodies. To see these descriptions in MATLAB command window, use command `help`, for example,

```
>> help EqnWeak3D
```

4 Properties of visualized sets

Let us denote by H the visualized solution set and by po_k (*piece in k-th orthant*) the intersection of H with the k-th orthant of $\mathbb{R}^3$, $k \in \{1, 2, \ldots, 8\}$.

A non-empty set $M \subseteq \mathbb{R}^n$ is said to be *bounded*, if there exists $\lambda \in \mathbb{R}$ such that, for each point x of the set M, the distance from x to the origin is not greater than λ. The empty set is considered as bounded. **Each set po_k, as H in whole, may be bounded or not depending on the input data of the corresponding system** (1)–(8).

By *polyhedron* in $\mathbb{R}^n$, we call a subset of $\mathbb{R}^n$ that may be represented as a solution set to a system of linear inequalities

$$Ax \geq b, \qquad A \in \mathbb{R}^{m \times n}, \ x \in \mathbb{R}^n, \ b \in \mathbb{R}^m, \ m, n \in \mathbb{N}.$$

A *polytope* is a bounded polyhedron, and a *polyhedral set* is a union of finite number of polyhedrons. Note that the space $\mathbb{R}^n$ is a polyhedron, while the empty set is a polytope. **All the sets po_k, $k = 1, \ldots, 8$, are polyhedrons. The set H is a polyhedral set.**

A set M is said to be *connected*, if any two points from it can be joined by a path lying in M. A *connected component* of the set is its connected subset that is maximal by inclusion. **Each po_k is connected because it is convex. The set H may be connected or disconnected and can have up to 8 connected components.** (H has 8 connected components if all po_k, $k = 1, \ldots, 8$, are nonempty and pairwise disjoint.)

Dimension of a nonempty *polyhedron* is dimension of its affine hull. The dimension of the empty set is assumed to be -1. We call a polyhedron *bodily*, if it has inner points, and *thin (or meager)* otherwise. **The dimension of a separate piece po_k of the set H may be from -1 (for the empty set) to 3 (for bodily polyhedron).**

The set H can have a cavity. The definition and examples of the sets with cavities are in Section 6.6.

A hyperplane P in $\mathbb{R}^n$ is named *supporting hyperplane* of a closed set M, if P and M have at least one common point, and M is contained in a closed half-space bounded by P. By *support* of a polyhedron M, we call the intersection of M with any its supporting hyperplane. It is obvious that supports of a polyhedron are polyhedrons too.

Each set po_k (as well as every polyhedron in $\mathbb{R}^3$) **can have supports whose dimensions are** 0, 1 **or** 2. The support of the dimension 0 is a *vertex*, the support of the dimension 1 is an *edge*, the support of the dimension 2 is a *face* of po_k.

We say that a point of $\mathbb{R}^3$ is an *orientation point* (*of H*), if it is a vertex of some po_k, $k \in \{1, 2, \dots, 8\}$. **The set** H **is empty if and only if it has no orientation points.**

5 Notation in figures

To work with figures of the set H in $\mathbb{R}^3$, we introduce the following definitions.

A *cut box* is a box (i.e. rectangular parallelepiped with edges parallel to the coordinate axes) such that its intersection with the set H is to be visualized by the package `IntLinInc3D`. There exist two types of cut boxes depending on the type of the cut: *automatic* cut box is calculated by the package `IntLinInc3D`, while *prescribed* cut box is inputted by the user.

Let us denote by $\widetilde{\mathrm{po}_k}$ the intersection of the set po_k with the cut box.

A *real face* is a face of $\widetilde{\mathrm{po}_k}$ lying on the boundary of the set H.

An *automatic cut face* is a face of $\widetilde{\mathrm{po}_k}$ lying on the boundary of the automatic cut box.

A *prescribed cut face* is a face of $\widetilde{\mathrm{po}_k}$ lying on the boundary of the prescribed cut box.

A *face from orthant* is a face of $\widetilde{\mathrm{po}_k}$ arising from the intersection of H with the k-th orthant. A face from orthant lies neither on the boundary of the set H nor on the boundary of the cut box.

Notation in figures: • is an orientation point,

🟩 is a real face,

🟥 is an automatic cut face,

🟨 is a prescribed cut face.

Faces from orthants are not visible in figures of the set H.

6   Recommendations
on preparation and analysis of figures

How to choose and run a launch function according to the system (1)–(8) and their solution types is explained in IntLinInc2D User manual. This is why such explanation is omitted here. We mention only that, in the case of three unknowns, the launch functions have input arguments `OrientPoints`, `transparency` and `varargin`, but do not have output arguments $P1, P2, P3, P4$.

The package `IntLinInc3D` provides users with the following information about the solution set H:

1) messages in command window, in particular the message about the number of orientation points;
2) the list of orientation points as output argument of the launch functions;
3) a figure of the solution set in a special graphics window.

The number of orientation points and their list are objective geometric characteristics of the solution set. They do not depend on what part of the solution set is visualized in the figure. We do not explain how to get the list of orientation points because this is standard MATLAB way of accessing the output arguments.

The situation with figures is much more complicated. In the case of two unknowns, the figure produced by the package `IntLinInc2D` gives full and explicit information about the geometric structure of the solution set. On the contrary, in the case of three unknowns, the figure produced by the package `IntLinInc3D` is only a rough draft for analysis of the geometric structure of the solution set and for getting the final figure. If someone wants

- to make the rough draft properly,
- to analyse the structure of the solution set relying on the rough draft
- and to prepare a final figure that gives exact representation of the geometric structure of the solution set and is suitable for saving as 2D image,

he should use not only standard MATLAB tools, but a special knowledge about the package `IntLinInc3D` too. Therefore recommendations on preparation and analysis of figures in `IntLinInc3D` package take the greater part of this User manual. The most important of them are marked with "**!**".

6.1 Tools for preparation and analysis of figures

A user of `IntLinInc3D` package has two toolkits for preparation and analysis of figures:

- Data Exploration Tools of MATLAB (Zoom, Rotate 3D, Scene Light, etc.);
- *view arguments* of the launch functions in `IntLinInc3D` package which include input arguments `OrientPoints` and `transparency` as well as optional input argument `varargin`.

The input argument `OrientPoints` is a parameter that controls drawing of the orientation points:

> if it has the value 0, the program does not plot these points;
> if it has the value 1, the orientation points are plotted.

The input argument `transparency` is a parameter responsible for the transparency of the real faces; its values may be 0 or 1:

> 0 means the absence of the transparency,
> 1 means that the real faces are transparent.

The optional input argument `varargin` allows a user to prescribe the cut box. This argument is inputted as 6 numbers, denoted as xb, xe, yb, ye, zb, ze. If such six numbers are present at the list of input arguments of the launch function, then the cut box is $[xb, xe] \times [yb, ye] \times [zb, ze]$, otherwise the package `IntLinInc3D` itself calculates the cut box.

At the first visualization of the solution set, it is necessary to assign the following values to the view arguments of the launch function: !

```
>> OrientPoints = 1;
>> transparency = 1;
```

the optional input argument `varargin` is absent. These are the *start values* of the view arguments. In the general case, only these values allow to receive full and explicit information about the geometric structure of the solution set. At the subsequent calls of the launch functions, you can change view arguments to get more good-looking figure (arguments `OrientPoints`, `transparency`) or to view its fragment (argument `varargin`).

6.2 How to make a diamond

Here, we describe a typical way of obtaining a good picture of the solution set.

Example 1 (Diamond)**.** We need to obtain a figure of the tolerable solution set for the system

$$\begin{pmatrix} 3.5 & [0,2] & [0,2] \\ [0,2] & 3.5 & [0,2] \\ [0,2] & [0,2] & 3.5 \end{pmatrix} x = \begin{pmatrix} [-1,1] \\ [-1,1] \\ [-1,1] \end{pmatrix}.$$

Sequence of actions.

1) *Inputting the initial data for the system.* In this example, we consider the system of equations (2) in which

$$\underline{A} = \begin{pmatrix} 3.5 & 0 & 0 \\ 0 & 3.5 & 0 \\ 0 & 0 & 3.5 \end{pmatrix}, \ \overline{A} = \begin{pmatrix} 3.5 & 2 & 2 \\ 2 & 3.5 & 2 \\ 2 & 2 & 3.5 \end{pmatrix}, \ \underline{b} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}, \ \overline{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

We input the data step by step:

```
>> infA = [ 3.5 0 0; 0 3.5 0; 0 0 3.5 ];
>> supA = [ 3.5 2 2; 2 3.5 2; 2 2 3.5 ];
>> infb = [ -1; -1; -1 ];
>> supb = [ 1; 1; 1 ];
```

2) *Calling the launch function with the start values of the view arguments.* To get a 3D draft picture that adequately represents the geometry of the solution set, we call the auxiliary function `EqnTol3D` with the start values of the view arguments:

```
>> OrientPoints = 1;
>> transparency = 1;
>> EqnTol3D(infA,supA,infb,supb,OrientPoints,transparency);
```

We get

```
Number of orientation points = 27
```


9
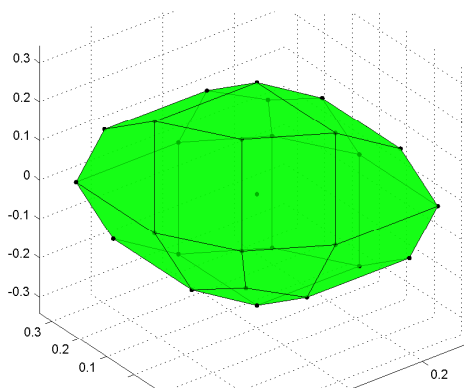
3) *Choosing the view arguments.* Analyzing the picture, we can conclude that the solution set is bounded (see Section 6.3), it does not have a cavity (see Section 6.6), and every nonempty set po_k is a bodily polyhedron. Such solution sets are better viewed without orientation points, while turning the transparency of the real faces off is a matter of taste.

Let us run the function `EqnTol3D` once again, but without visualizing the orientation points and without transparency of the real faces:

```
>> EqnTol3D(infA,supA,infb,supb,0,0);
```

The new picture looks like this. $\longrightarrow$
At the picture, the solution set
resembles a cobblestone.
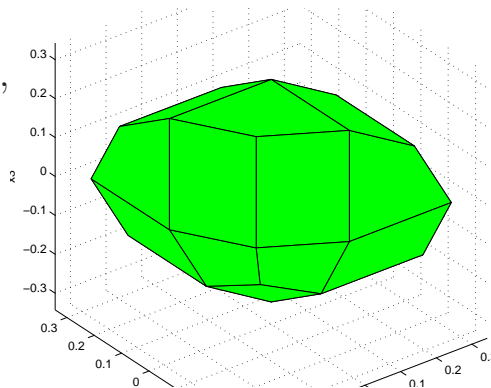
4) *Processing the picture.* To make a diamond of the cobblestone, we use the standard Data Exploration Tools from MATLAB:
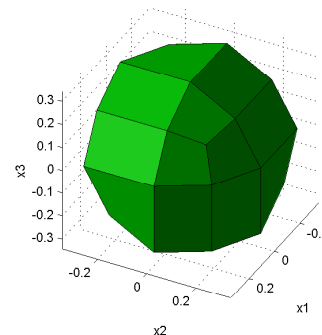
we rotate (using tool Rotate 3D),   decrease the size (using tool Zoom)
$\downarrow$ $\downarrow$
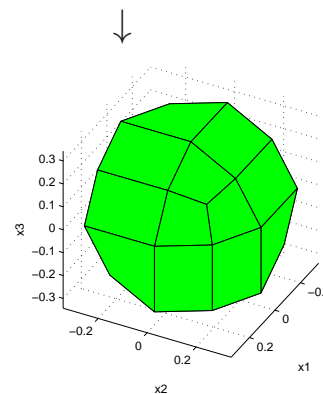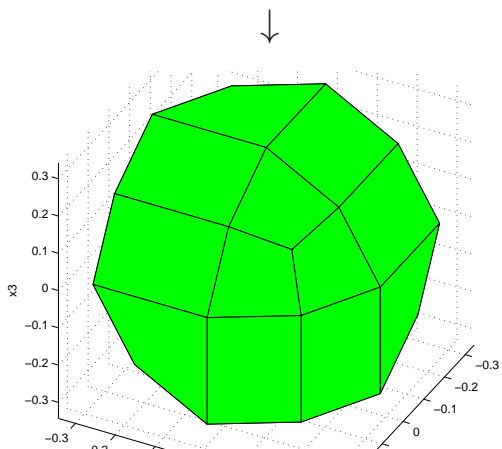
and turn on the light (using Scene Light). $\longrightarrow$
The obtained picture is suitable for saving as
2D image in `eps`, `pdf`, `jpeg` and such like formats.

Remark. In the text below, all pictures have been processed by MATLAB Data Exploration Tools after being outputted from `IntLinInc3D` package.

6.3 Unbounded sets

The present subsection is devoted to two tools of the package `IntLinInc3D`
designed for creating adequate pictures of unbounded sets. These tools are
automatic cut and plotting orientation points.

6.3.1 Automatic cut

A standard trick that should be employed in the course of visualizing unbounded
sets by computer systems is to prescribe the range of coordinates (or, in other
words, axis limits). As the result, we only see a part of the set bounded by the
specified range.

The package `IntLinInc3D` is also able to visualize a part of the solution
set within a predefined coordinate range. We refer to such a coordinate range
as *prescribed cut box*. It is defined by an additional argument `varargin`. In
pictures, the faces that arise from the prescribed cut are transparent and have
yellow color in contrast to real faces of the solution set that have green color
and varied transparency.

The standard trick has two drawbacks:
1) the user himself must carry out a preliminary analysis of the set under
   visualization and choose a suitable coordinate range,
2) the picture within a predetermined coordinate range does not allow one
   to make definite conclusions on global properties of the set, in particular,
   on whether it is bounded or unbounded.

The advantage of the package `IntLinInc3D` in visualization of unbounded
sets is that one does not need to specify the prescribed cut box. If the pre-
scribed cut box is not inputted, then the package `IntLinInc3D` itself chooses
the coordinate range for drawing the solution set, and produced picture con-
tains all the essential information about the solution set. We refer to such a
coordinate range as *automatic cut box*. In case of automatic cut, the package
`IntLinInc3D` takes the following actions:
- finds the set of all orientation points,
- chooses the cut box wider than the interval hull
  of the set of orientation points,
- draws intersection of the solution set with the cut box.

At the picture, the faces arising from intersection of unbounded solution set with the automatic cut box are always transparent and have red color.

Unlike the prescribed cut, the automatic cut does not require a preliminary work of the user and preserves information about global properties of the solution set.

Next, we give recommendations that will provide the user with capability to distinguish between bounded and unbounded solution sets judging on the picture obtained by automatic cut.

1) First of all, we should note that the problem of recognition whether H is (un)bounded can be reduced to the problem of recognition whether the sets po_k are (un)bounded. The set H is bounded if all po_k, $k = 1, 2, \ldots, 8$, are bounded. If at least one of the components po_k is unbounded, then H is unbounded too. !

2) If the set po_k has a red face, then it is unbounded. !

Example 2. The united solution set for the system of equations

$$
\begin{pmatrix}
[-1, 1] & 0 & 0 \\
0 & [-1, 1] & 0 \\
0 & 0 & [-1, 1]
\end{pmatrix} x =
\begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix}
$$

is unbounded and consists of 8 trihedral angles. To get the picture of this set, input

```
infA = [ -1 0 0; 0 -1 0; 0 0 -1 ];
supA = [  1 0 0; 0  1 0; 0 0  1 ];
infb = [ 1; 1; 1 ];
supb = [ 1; 1; 1 ];
EqnWeak3D(infA,supA,infb,supb,0,1);
```
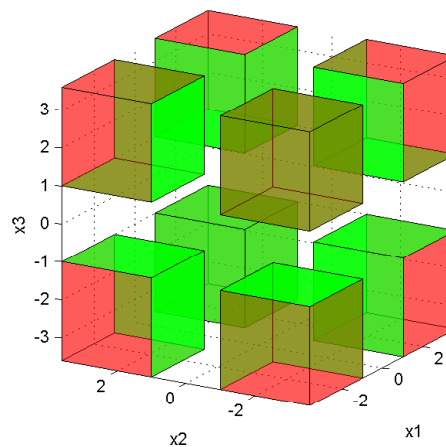
3) If the set po$_k$ is bodily and all its faces have green color, then the set po$_k$ is bounded.

Example 3. The united solution set to the system of equations

$$\begin{pmatrix} [-1,1] & 0 & 0 \\ 0 & [-1,1] & 0 \\ 0 & 0 & [-1,1] \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ [-2,2] \\ [-2,2] \\ [-2,2] \end{pmatrix}$$

is bounded and consists of 8 cubes.
To produce its picture, input the commands

```
infA = [ -1 0 0; 0 -1 0; 0 0 -1; eye(3) ];
supA = [  1 0 0; 0  1 0; 0 0  1; eye(3) ];
infb = [ 1; 1; 1; -2; -2; -2 ];
supb = [ 1; 1; 1;  2;  2;  2 ];
EqnWeak3D(infA,supA,infb,supb,0,1);
```
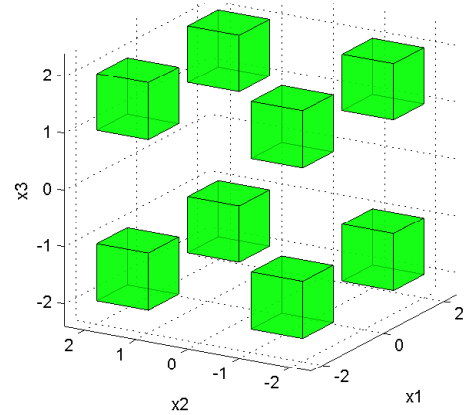

4) Specification of the optional argument `varargin` often prevents from determining whether the solution set is (un)bounded.

Example 4. If, in Examples 2 and 3, we take the prescribed cut box as $[-1.5, 1.5] \times [-1.5, 1.5] \times [-1.5, 1.5]$ and call the package by the command

```
EqnWeak3D(infA,supA,infb,supb,0,1,-1.5,1.5,-1.5,1.5,-1.5,1.5);
```

the pictures will be identical:


13
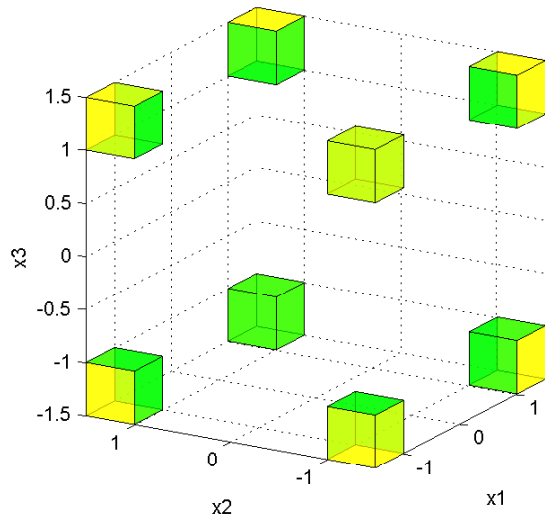
6.3.2 Plotting orientation points

In Examples 2 and 3, we intentionally assigned the value 0 to the argument `OrientPoints` in order to show that sometimes (when e.g. all the nonempty sets po_k are bodily) we can do without drawing orientation points. In the general case, only combination of the automatic cut with plotting orientation points produces a **boundedness criterion for the set** po_k: The set po_k is bounded if and only if, at the picture obtained with the automatic cut and plotting orientation points, either the set po_k does not have edges or two vertices are marked at every its edge.

Example 5. The united solution set to the system of relations

$$
\begin{pmatrix}
[-1,1] & 0 & 0 \\
0 & [-1,1] & 0 \\
0 & 0 & [-1,1] \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
x
\begin{pmatrix}
= \\
= \\
= \\
= \\
= \\
\geq
\end{pmatrix}
\begin{pmatrix}
1 \\
1 \\
1 \\
[-1,2] \\
[-1,2] \\
-2
\end{pmatrix}
$$

consists of bounded (a segment, two squares, a cube) and unbounded pieces (a ray, two half-strips and a semi-infinite square prism). One can get the picture using the commands

```
infA = [ -eye(3); eye(3) ];
supA = [  eye(3); eye(3) ];
infb = [ 1; 1; 1; -1; -1; -2 ];
supb = [ 1; 1; 1;  2;  2; -2 ];
relations=['=';'=';'=';'=';'=';'>'];
MixWeak3D(infA,supA,infb,supb,relations,1,1);
```
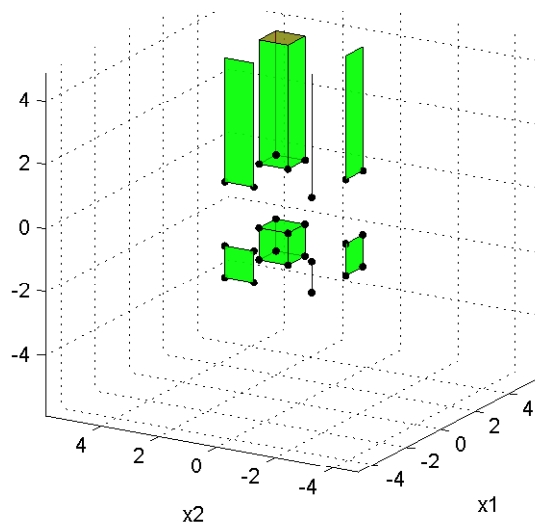

To sum up, in order to determine boundedness of the solution set from its picture, we have to permit the package (i) to choose the cut box automatically (to achieve this, it is sufficient just not to specify the argument `varargin`) and (ii) to draw the orientation points (assign 1 to the argument `OrientPoints`).

14

6.4 Thin (meager), but important

To ensure that, at the picture created by the package `IntLinInc3D`, thin polyhe- !
drons po_k are shown correctly and unambiguously, the argument `OrientPoints`
must have the value 1. Otherwise, we will not see isolated points that are sep-
arate connected components of the set H, and, also, we will not be able to
distinguish bounded meager sets po_k from unbounded ones (a segment from a
ray or a straight line, a plane angle from a triangle, and so on).

Example 6. The united solution set to the system of equations

$$
\begin{pmatrix}
[-1,1] & 0 & 0 \\
0 & [-1,1] & 0 \\
0 & 0 & [-1,1] \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix} x =
\begin{pmatrix}
1 \\
1 \\
1 \\
[-1,2] \\
[-1,2] \\
[-1,2]
\end{pmatrix}
$$

is composed of a point, three segments, three squares and a cube. At the left
picture, the point is not visible, and the segments and squares are depicted so
that their boundedness is doubtful. At the right-hand picture, all the compo-
nents of the solution set are depicted correctly and interpreted unequivocally.
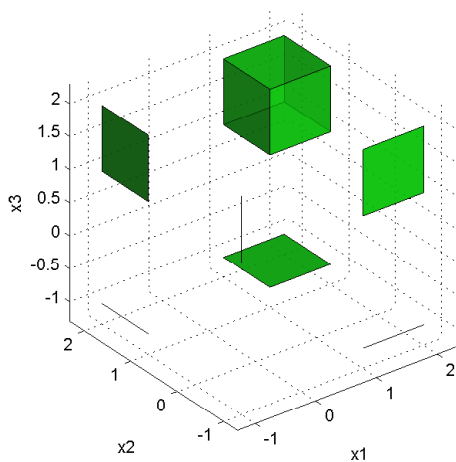
OrientPoints $= 0$ OrientPoints $= 1$

(Use Example 3 to input the data and to run the package.)

15
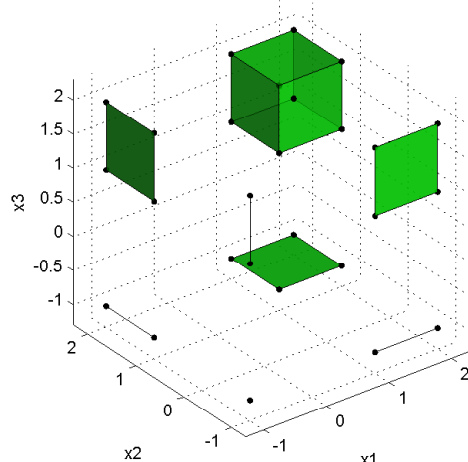
6.5 Empty set and whole space

In this section, we consider how one can conclude that the solution set is empty or it coincides with the whole space $\mathbb{R}^3$ after examination of the pictures and output produced by `IntLinInc3D`.

6.5.1 Empty set

Emptiness of the solution set H means that all its intersections po_k, $k = 1, \ldots, 8$, with separate orthants are empty. Every polyhedron po_k does not contain straight lines, therefore its emptiness is equivalent to the absence of vertices. Overall, the set H is empty if and only if it does not have orientation points. When applied to the work of the package `IntLinInc3D`, the above theoretical statement transforms to the following recommendation on how to recognize emptiness of the set H.

The solution set is empty if and only if the package produces no picture and ! outputs the message

```
Number of orientation points = 0
The solution set is empty
```

The above recommendation does not depend on the specific values of the view arguments for which the launch function runs, but relies on the fact that the package correctly "understands" the geometry of the solution set. This is also true for the other recommendations of the manual.

Example 7. It is obvious that the equation $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix} x = 1$ does not have solutions $x \in \mathbb{R}^3$. To see how the package `IntLinInc3D` processes it, just type

```
uC = [ 0 0 0 ];
oC = uC;
ud = 1 ;
od = ud;
Cxind3D(uC,oC,ud,od,0,1);
```
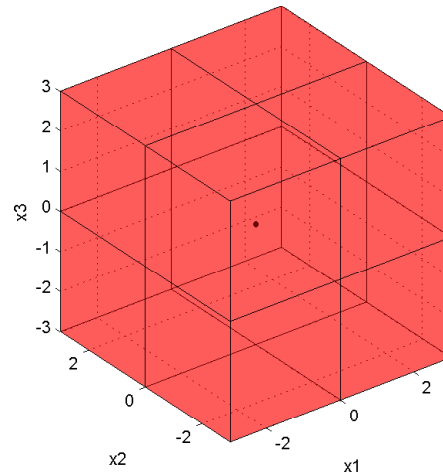
6.5.2 Whole space

The solution set coincides with the whole space $\mathbb{R}^3$ if and only if the call of   !
the package without the optional argument `varargnin` results in the output
message

```
Number of orientation points = 1
```

and transparent red cube as a picture. For `OrientPoints = 1`, the origin of
coordinates is marked as an orientation point. The value of the argument
`transparency` does not affect the picture.
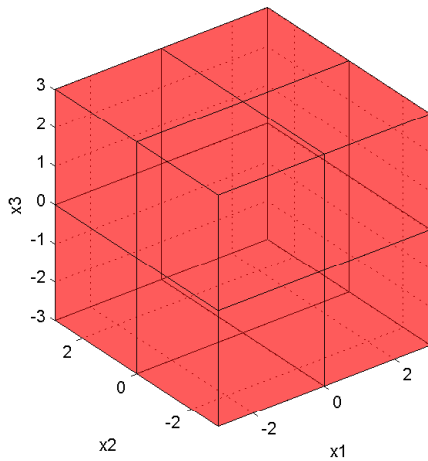
$$\mathtt{OrientPoints} = 0 \qquad\qquad \mathtt{OrientPoints} = 1$$

Example 8. The united solution set to the interval inequality

$$\big([-1,1]\ \ [-1,1]\ \ [-1,1]\big)\, x \geq 0$$

is the whole space. One can obtain its various pictures by inputting

```
infA = [ -1 -1 -1 ];
supA = [  1  1  1 ];
infb = [ 0 ];
supb = [ 0 ];
GeqWeak3D(infA,supA,infb,supb,0,0);
```

and varying the values of the last two arguments of the function `GeqWeak3D`.

6.5.3   And what is it?

There are two pictures that can be mistakenly interpreted as either empty solution set or a solution set coinciding with the whole space. These are "empty white box" (coordinate system without any objects) and "empty yellow box".

<center>What does "empty white box" mean?</center>

First of all, one should bear in mind that the "empty white box" in no way   **!**
means that the solution set is empty or that the solution set coincides with the whole space.

If, at the picture, you see a coordinate system that does not have visualized objects, then you should pay attention to the arguments `varargnin` and `OrientPoints`. The argument `varargnin` determines a prescribed cut box for the solution set. Therefore, when the package is called with the optional argument `varargnin`, the picture contains only a part of the solution set within this box. The arguments `OrientPoints` helps either displaying or hiding the orientation points of the solution set. Depending on the specific values of these arguments, there exist three ways of correct interpretation of the "empty white box".

1) If the argument `varargnin` is present and `OrientPoints` $= 1$, then the "empty white box" means that the completion of the solution set contains the entire prescribed cut box.

Example 9. In the Diamond example (see Section 5.2, Example 1), let us specify the prescibed cut box $[1, 2] \times [1, 2] \times [1, 2]$ and set `OrientPoints` equal to 1 in the arguments of the function `EqnTol3D`:

```
>> EqnTol3D(infA,supA,infb,supb,1,0,1,2,1,2,1,2);
```

We get the picture $\longrightarrow$

<center>18</center>
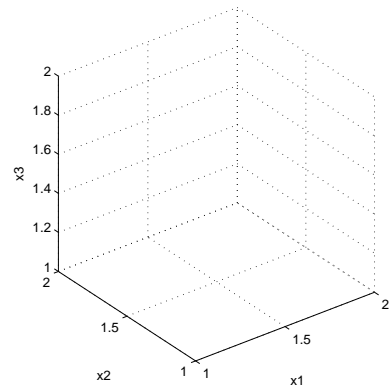
2) If the argument `varargnin` is present and `OrientPoints` = 0, then set `OrientPoints` = 1 and rerun the package to make sure that there exit (or do not exist) isolated points of the solution set in the prescribed cut box.

3) If the argument `varargnin` is not present, then the "empty white box" means that the argument `OrientPoints` = 0 and the solution set consist of isolated points. For correct visualization of the solution set, one should call the package with `OrientPoints` = 1.
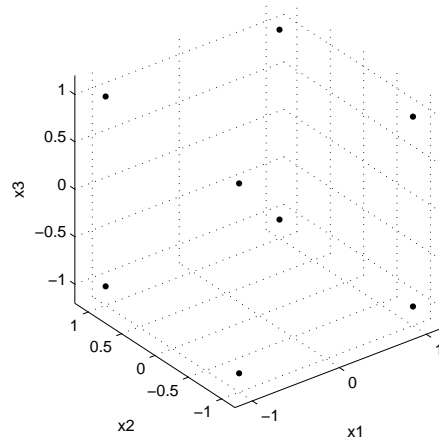
Example 10. The united solution set to the equation

$$
\begin{pmatrix}
[-1,1] & 0 & 0 \\
0 & [-1,1] & 0 \\
0 & 0 & [-1,1] \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
x =
\begin{pmatrix}
1 \\
1 \\
1 \\
[-1,1] \\
[-1,1] \\
[-1,1]
\end{pmatrix}
$$

is comprised of eight points, but for `OrientPoints` = 0 it looks like the "empty white box":
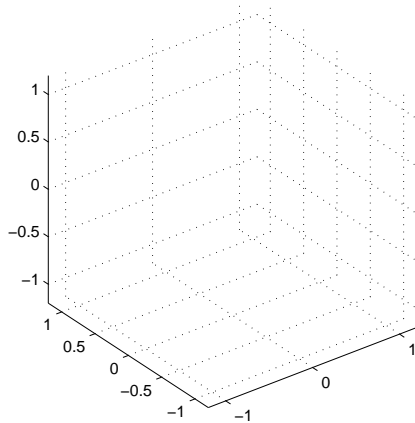
OrientPoints = 0   OrientPoints = 1

(Use Example 3 to input the data and to run the package.)

We emphasize that the "empty yellow box" never means that the solution ！
set is empty or that the solution set coincides with the whole space.

The "empty yellow box" at a picture indicates that the package has been
called with the optional argument `varargnin`, and the prescribed cut box from
this argument lies entirely in the solution set. The origin of coordinates in the
"empty yellow box" may be marked as an orientation point (when it belongs
to the prescribed cut box and `OrientPoints = 1`).

Example 11. The solution set of the two-sided inequality $-5 \leq x_1 \leq 5$
includes the entire box $[-3, 3] \times [-3, 3] \times [-3, 3]$. Let us input the initial data

```
uC = [ 1 0 0 ];
oC = uC;
ud = -5;
od = 5;
```

and call the function `Cxind3D` with the prescribed cut box $[-3, 3] \times [-3, 3] \times$
$[-3, 3]$ specified in `varargnin`. Depending on the values of the argument
`OrientPoints`, we obtain the pictures
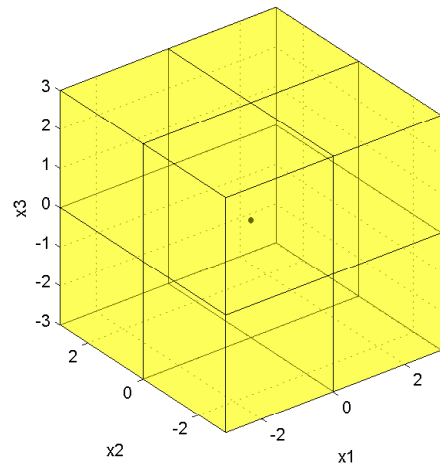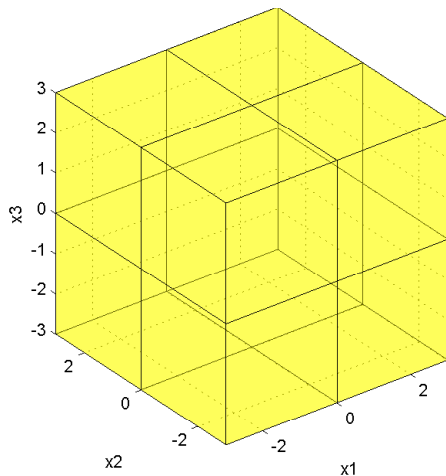

```
Cxind3D(uC,oC,ud,od,0,0,-3,3,-3,3,-3,3);    Cxind3D(uC,oC,ud,od,1,0,-3,3,-3,3,-3,3);
Cxind3D(uC,oC,ud,od,0,1,-3,3,-3,3,-3,3);    Cxind3D(uC,oC,ud,od,1,1,-3,3,-3,3,-3,3);
```

In this example, the "empty yellow box" looks very much like the picture of
the whole space $\mathbb{R}^3$ and differs only in color.

6.6 How to test the presence of cavity

We shall speak that the solution set H has a *cavity*, if the origin of coordinates does not lie in H, while every ray starting from the origin of coordinates intersects H.

Any cavity is a bounded polyhedral set. It is not necessarily convex, but it is always star-convex with respect to the origin of coordinates (for every point x from the set, it also contains the segment $[0,1]x$). As a consequence, the cavity is connected. The origin of coordinates is an interior point of the cavity. Figuratively speaking, the cavity is a "house" for the origin of coordinates, its walls built from real faces of the solution set, and there are no doors and windows. In this section, we discuss how to tune the view arguments to test the presence of a cavity from the information provided by the picture.

1) Using the automatic cut (i.e., absence of the optional argument `varargin`), even for any arbitrary values of the arguments `OrientPoints` and `transparency`, makes it evident whether the cavity is present or not for a sufficiently wide class of the solution sets.
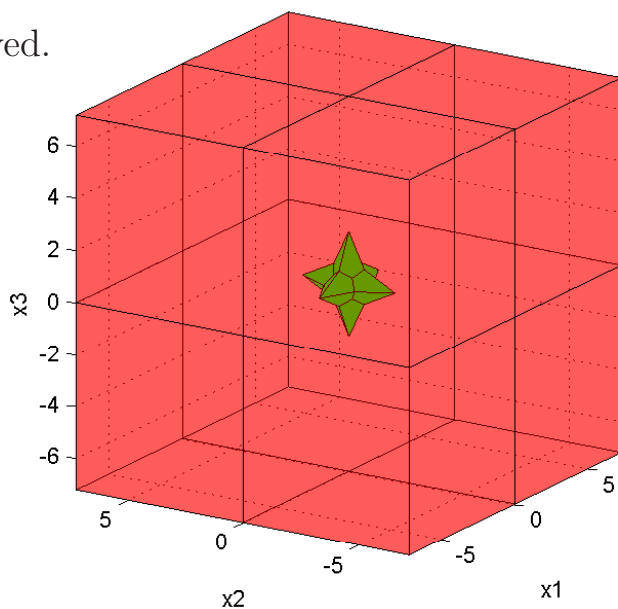
Example 12 (Whole space without a star). Controllable solution set to the system of interval equations

$$\begin{pmatrix} [-1,1] & [-2,2] & [-2,2] \\ [-2,2] & [-1,1] & [-2,2] \\ [-2,2] & [-2,2] & [-1,1] \end{pmatrix} x = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$$

is the whole space $\mathbb{R}^3$ with a star removed. To obtain its picture, input the commands

```
supA = [1 2 2; 2 1 2; 2 2 1];
infA = -supA;
infb = 2*ones(3,1);
supb = infb;
EqnCtl3D(infA,supA,infb,supb,0,0);
```


21

2) If the automatic cut is supplemented by transparency of the real faces (i.e., `transparency = 1`), this substantially extends the class of the solution sets for which one can determine, from the picture, either presence or absence of the cavity.

Example 13 (Cube without mirror pyramid). The united solution set to the system of interval equations

$$\begin{pmatrix} [-1,1] & [-1,1] & [-1,1] \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 0.5 \\ [-1,1] \\ [-1,1] \\ [-1,1] \end{pmatrix}$$
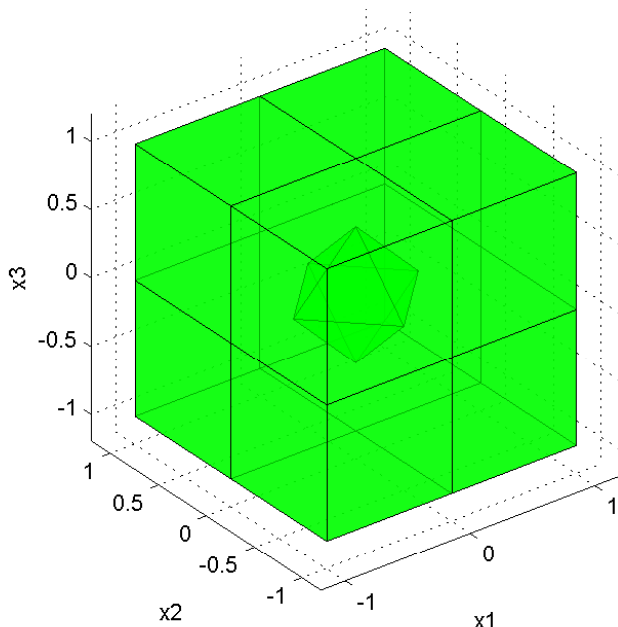
is a cube with the "mirror pyramid" deleted. In order to evaluate the influence of the argument `transparency`, input the commands

```
infA = [-ones(1,3); eye(3)];
supA = [ones(1,3); eye(3)];
infb = [.5; -ones(3,1)];
supb = [.5; ones(3,1)];
EqnWeak3D(infA,supA,infb,supb,0,0);
```

Then change the last argument of the function `EqnWeak3D` from 0 to 1 and compare the pictures obtained:
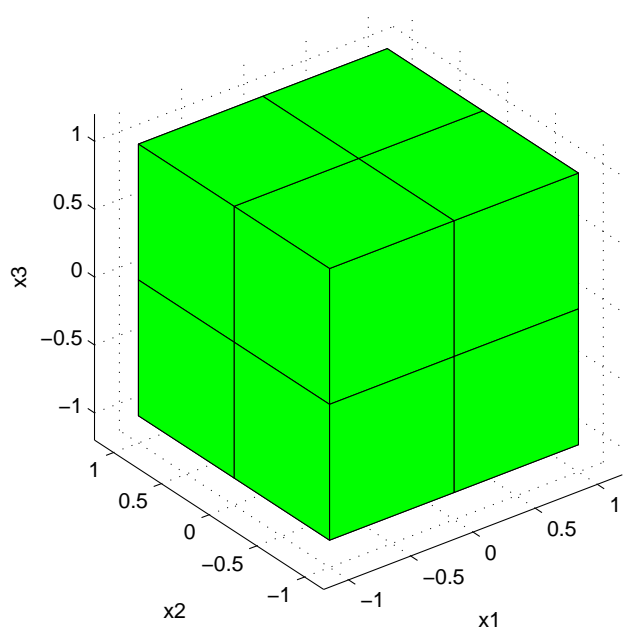
transparency $= 0$ transparency $= 1$

3) Using the start values of all view arguments enables one to uniquely determine, from the picture, the presence or absence of the cavity for almost every solution set.

In particular, the following rule may prove helpful: if, for the start values of the view arguments, the origin of coordinates is marked as an orientation point at the picture, then the solution set has no cavity.

Example 14 (Mirror pyramid)**.** The tolerable solution set for the interval equation

$$\left([-1,1] \quad [-1,1] \quad [-1,1]\right) x = \left([-1,1]\right)$$

is a "mirror pyramid":

```
infA = [-ones(1,3)];
supA = [ones(1,3)];
infb = -1 ;
supb =  1 ;
EqnTol3D(infA,supA,infb,supb,1,1);
```
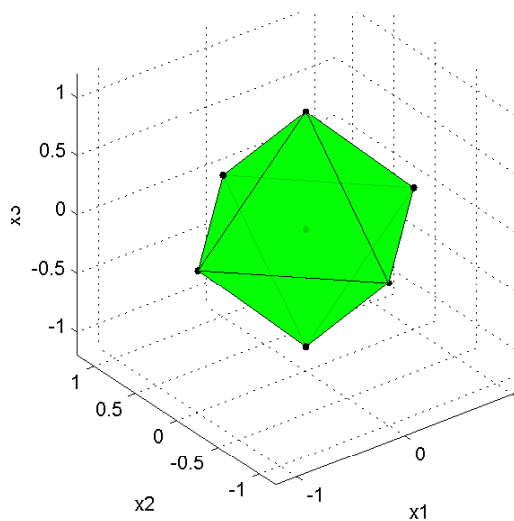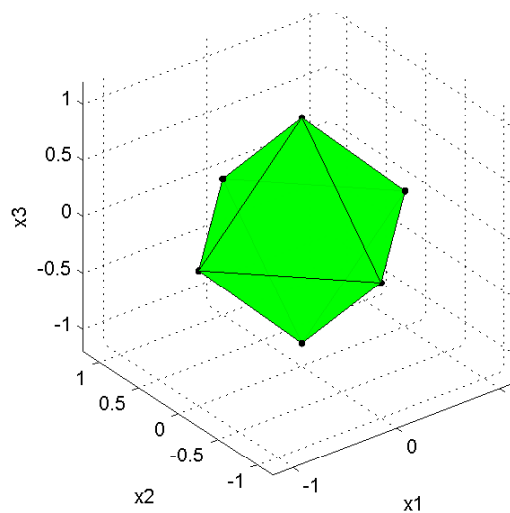

Example 15 (Boundary of mirror pyramid)**.** AE-solution set to the system

$$\begin{pmatrix} [-1,1]^\forall & [-1,1]^\forall & [-1,1]^\forall \\ [-1,1]^\exists & [-1,1]^\exists & [-1,1]^\exists \end{pmatrix} x = \begin{pmatrix} [-1,1]^\exists \\ 1^\exists \end{pmatrix}$$

is the boundary of a "mirror pyramid":

```
infA = [-ones(2,3)];
supA = [ones(2,3)];
infb = [ -1; 1 ];
supb = [  1; 1 ];
Aq = [ 'A' 'A' 'A'; 'E' 'E' 'E' ];
bq = [ 'E' ; 'E' ];
EqnAEss3D(infA,supA,Aq,infb,supb,bq,1,1);
```


23

4) Finally, we have the optional argument `varargin` that enables one to visualize intersection of the solution set H with any desired box.

The prescribed cut box, determined by the argument `varargin`, allows to see the intersection of the solution set with any separate orthant $\mathcal{O}_k$. To do this, we

1) put one of the box vertices in the origin of the coordinates,
2) put the opposite vertex of the box into the interior of the orthant $\mathcal{O}_k$ at such a distance that the box contains, with suitable excess, all orientation points from the orthants $\mathcal{O}_k$ (the information about orientation points of the solution set within an orthant can be extract from the picture produced for the start values of the view arguments or from the list of orientation points),
3) assign the chosen box to the optional argument `varargin`.

If the picture obtained for the start values of the view arguments does not allow one to definitely conclude whether there is/is not a cavity in the solution set, we recommend the following way of actions that will resolve this uncertainty. In the launch function, set $\texttt{OrientPoints} = 1$ and vary the value of the optional argument `varargin` to look at the intersection of the solution set with each orthant. The solution set H has a cavity if and only if, for each $k = 1, \ldots, 8$, at the picture of the intersection of the solution set H with the orthant $\mathcal{O}_k$,

- the origin of coordinates is not marked as an orientation point
- and every coordinate axis contains an orientation point.

Example 16. In Examples 12–15, every column of the matrix is symmetric with respect to zero (i.e., is a balanced interval vector), therefore, the solution set is symmetric with respect to every coordinate plane. To check whether such a set has a cavity or not, it suffices to see its intersection with only one orthant. For definiteness, we take the positive orthant. In the launch function for each of Examples 12–15, we set $\texttt{OrientPoints} = 1$ and specify a suitable prescribed cut box `varargin`:
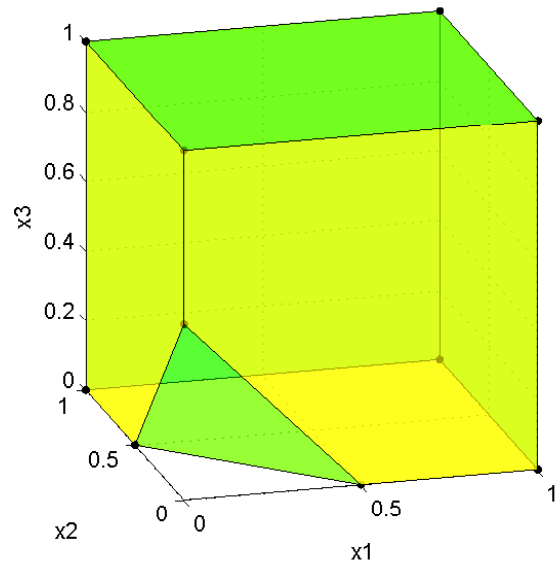
Whole space without a star

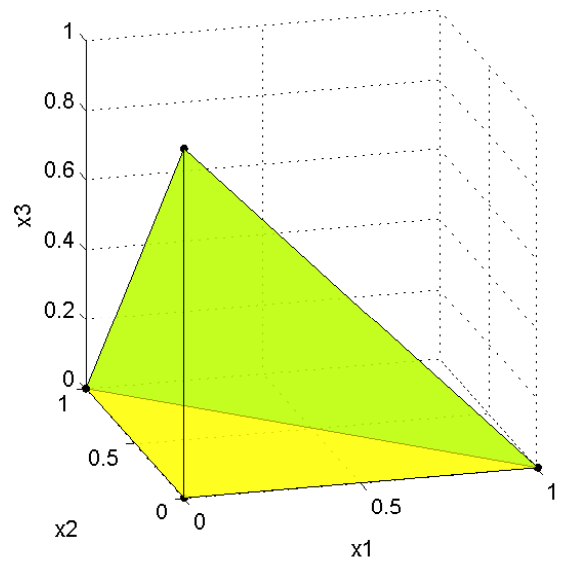`EqnCtl3D(infA,supA,infb,supb,1,0,0,3,0,3,0,3);`
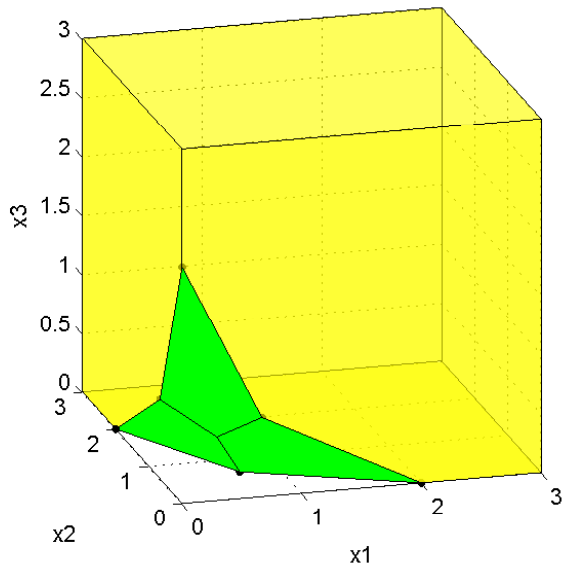
Cube without mirror pyramid

`EqnWeak3D(infA,supA,infb,supb,1,1,0,1,0,1,0,1);`
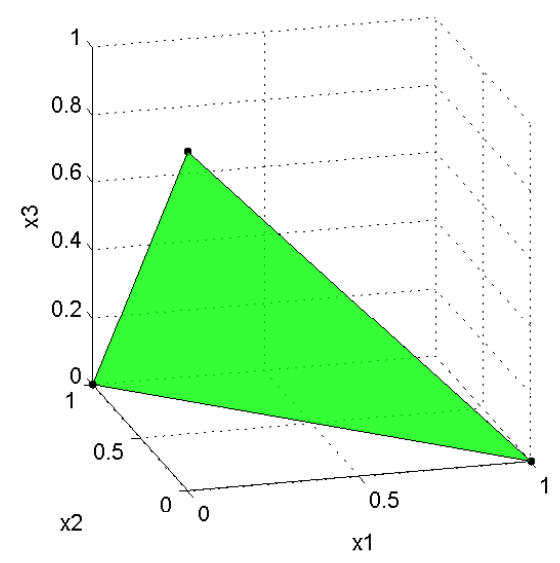

Mirror pyramid

`EqnTol3D(infA,supA,infb,supb,1,1,0,1,0,1,0,1);`

Boundary of mirror pyramid

`EqnAEss3D(infA,supA,Aq,infb,supb,bq,1,1,0,1,0,1,0,1);`

It is obvious from the pictures that every solution set, except the mirror pyramid, has a cavity.

7 How to install and operate the package `IntLinInc3D`

1. Download the file
   http://interval.ict.nsc.ru/Programing/MCodes/IntLinInc3D.zip

2. Unpack it into a separate directory (folder).

3. Set MATLAB paths to this directory.

4. In MATLAB command window, input the initial data for the systems (1)–(8) and call launch functions according to this manual and to `IntLinInc2D` User manual.

8 References

[1] IRENE A. SHARAYA, Boundary intervals method for visualization of polyhedral solution sets, *Reliable Computing*, 19 (2015), No. 4, pp. 435-467.
http://www.nsc.ru/interval/sharaya/Papers/ReliableComputing-19-pp-435-467.pdf

[2] I. A. SHARAYA, Quantifier-free descriptions for interval-quantifier linear systems, *Trudy Instituta Matematiki i Mekhaniki UrO RAN* [*Proceedings of the Institute of Mathematics and Mechanics, Ural Branch of the Russian Academy of Sciences*], 20 (2014), No. 2, pp. 311–323. (in Russian)
http://interval.ict.nsc.ru/sharaya/Papers/trIMM14.pdf

[3] S. P. SHARY, A new technique in systems analysis under interval uncertainty and ambiguity, *Reliable Computing*, 8 (2002), No. 5, pp. 321–418.
http://interval.ict.nsc.ru/shary/Papers/ANewTech.pdf

[4] J. ROHN, Solvability of systems of interval linear equations and inequalities. In: *Linear optimization problems with inexact data*, M. Fiedler, J. Nedoma, J. Ramik, J. Rohn, K. Zimmermann. New York, Springer, 2006. P. 35–77.
http://interval.ict.nsc.ru/Library/InteBooks/InexactLP.pdf

Irene A. Sharaya   August 10, 2015
Institute of Computational Technologies SB RAS
Novosibirsk, Russia