VINCI version 1.0.5

Benno Büeler and Andreas Enge
Current maintainer:
Andreas Enge
Laboratoire d'Informatique
École polytechnique
91128 Palaiseau
France
enge@lix.polytechnique.fr

July 2003

1 Introduction

The programme VINCI implements several algorithms for computing the volume of a full dimensional bounded polyhedron (polytope). The polytope must be given by its vertex or hyperplane or double representation in the file format specified by David Avis and Komei Fukuda. For details on this format, see Section 3. The code is a byproduct of our research on efficient algorithms for polytope volumes; the implemented algorithms are described and analysed in [1].

Most of the algorithms are implemented directly in the code and can be used without installing any further programme. They need the hyperplane representation or the double description of the polytope. Some other methods perform calls to external programmes, which have to be installed separately, see Section 2.

VINCI can be distributed freely under the GNU GENERAL PUBLIC LICENSE. Please read the file `COPYING` carefully before using it. We cannot

accept any warranty for problems you might encounter with this package; be aware of using it at your own risk. However, comments and suggestions for improvements are always welcome. If you find VINCI useful, please tell us to which purpose you use it, and let us know about your experiences. Remember that the most powerful support for free software development is user's appreciation and collaboration.

2 Installation

VINCI is available from the following site:

```
http://www.lix.polytechnique.fr/Labo/Andreas.Enge/Vinci.html
```

After downloading the package, unpack it by typing

```
% gunzip vinci-1.0.5.tar.gz
% tar xvf vinci-1.0.5.tar
```

The package consists of the files listed in Table 1; they will be placed in the newly created subdirectory `vinci-1.0.5`.

Before using the programme please read the file `COPYING`. For changes to previous versions, consult the file `ChangeLog`. If you use the gnu C compiler `gcc` you should just have to type

```
% make
```

and the executable `vinci` will be created.

Otherwise edit the makefile and replace in the line `CC=gcc` the word `gcc` by the name of your local C compiler, typically `cc`. If nothing works, try

```
% cc -o vinci vinci*.c
```

The package is written in ANSI C and therefore should compile in any environment.

`vinci.h`	The header file for all .c-files
`vinci_global.c`	Global variable declarations
`vinci_memory.c`	Various routines handling dynamic memory allocation
`vinci_file.c`	Functions for reading the polyhedra files
`vinci_screen.c`	Functions for outputting control information on the screen
`vinci_set.c`	Implementation of set routines
`vinci_computation.c`	The main computational routines
`vinci_volume.c`	Most of the volume computation routines
`vinci_lass.c`	Lasserre's volume formula
`vinci.c`	The main programme
`makefile`	
`COPYING`	GNU GENERAL PUBLIC LICENSE
`ChangeLog`	A list of changes from the previous versions
`manual.tex`	The reference manual
`html.sty`	File needed for processing the manual
`square.ext`	A very simple example to get started
`square.ine`	

Table 1: Files in the VINCI distribution

If you want to use all of the volume computation algorithms you need to install the additional programme LRS, which can be obtained from the following site:

author:	David Avis (http://www.cs.mcgill.ca/~avis)
http site:	`http://cgm.cs.mcgill.ca/~avis/C/lrslib/`
file name:	`lrslib-*.tar.gz`, where "*" stands for the actual version

Please use LRSLIB in version 0.40 or later. After compiling the programme copy the executable `lrs` to the directory `vinci-1.0.5`.

3 Input Formats

3.1 Description

Generally spoken, a polytope can either be defined as the bounded intersection of finitely many halfspaces (we speak about *H-representation* in this case) or as the convex hull of its finally many vertices (*V-representation*). Depending on the specified algorithm, one or both of these description are needed. The polytope is communicated to the programme in the form of files using the polyhedra format of Avis and Fukuda. Different types of description are recognized from the default file name extension:

`.ine`: inequalities defining the polytope as intersection of halfspaces

`.ext`: vertices (extreme points) of the polytope

To describe the polytope $\{x \in \mathbb{R}^d : Ax \leq b\}$ where A is a matrix of dimension $m \times d$ and b a vector of dimension m, the corresponding `.ine`-file is given by:

---

various comments
begin
m $d+1$ **numbertype**
b $-A$
end
various options

`numbertype` can be one of `integer`, `float` or `rational`. In the case of rational number type each number can be given in ordinary integer representation or as a fraction p/q or $-p/q$, where p and q are positive integers of arbitrary length. At the time being, all rational input is converted to floating point, and all computations are done with floating point arithmetics. To facilitate conversion from H- to V-representation, it is recommended to write `H-representation` before `begin`.

To describe a polytope as the convex hull of its vertices $v_1, \ldots, v_n$, the format of the `.ext`-file is the following:

---

various comments
begin
n $d+1$ **numbertype**
1 v_1
$\vdots$ $\vdots$
1 v_n
end
various options

To facilitate conversion from V- to H-representation, it is recommended to write `V-representation` before `begin`.

3.2 Example

To illustrate the file format, let us consider the simple example of the square $\{(x, y) \in \mathbb{R}^2 : -1 \leq x, y \leq 1\}$. The file `square.ine` is given by:

```
square
H-representation
begin
4   3   integer
1   1   0
1   -1  0
1   0   1
1   0   -1
end
```

---

where the first two lines represent the inequalities for x and the last ones those for y.

The file `square.ext` presents itself as:

```
extreme points of a square
V-representation
begin
4   3   integer
1   -1  -1
1   -1   1
1    1  -1
1    1   1
end
```

3.3 Conversion between Representations

To make use of different volume computation algorithms which need different types of files, it may be desirable to change between V- and H-representation of a polytope. This job can be done using CDDLIB, see `http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`.

To convert the floating point file `square.ext` to `square.ine`, just type:

```
% cddf+ square.ine
```

The other way is performed by

```
% cddf+ square.ext
```

If you desire to perform a conversion with exact arithmetics for rational data, use `cddr+` instead of `cddf+`.

4 Running the Programme...

Several volume computation algorithms have been implemented which are more or less efficient on different problem classes. The way you use the programme will therefore depend on the knowledge you have about special properties of your problem.

Basically you can specify a certain method yourself or leave this task to the programme which will try to determine a hopefully efficient method.

4.1 ... Without Specifying a Method

To compute the volume of a polytope you do not have any special knowledge about just type `vinci` followed by the basic file name, i.e. without the file extensions which are appended automatically. The programme will then choose a method automatically as described in Section 4.4. In the example of Section 3.2 you would have to type

```
% vinci square
```

4.2 ... Specifying a Method

If your problem has a special structure some algorithm may be more efficient than another one although this is not detected by the automatic method proposal. So you might like to specify the method yourself. In this case use the option `-m` followed by the abriged name of the method. Thus

```
% vinci square -m rch
```

or

```
% vinci -m rch square
```

would use Cohen and Hickey's triangulation method to compute the square area.

Table 2 presents the implemented algorithms together with the files and the additional programmes needed. Please make sure that you have created the necessary files and installed the listed packages before choosing a method.

If you choose a method which cannot be used because files or programmes are missing you get a list of possible algorithms and a method proposal as described in Section 4.4.

Usually `hot` is the fastest code.

method	needed files	additional software	remarks
`hot`	`.ext`	—	uses a Cohen&Hickey–like face enumeration scheme and Householder orthogonalisation
`rlass`	`.ine`	—	Lasserre's method; specially suited for near–simple polytopes
`rch`	`.ext`	—	Cohen&Hickey–triangulation; specially suited for near–simplicial polytopes, numerically very robust
`lawd`	`.ine`	LRS	Lawrence's formula in the general case, numerically unstable
`lawnd`	`.ext` `.ine`	—	Lawrence's formula, only for non–degenerate (simple) polytopes; extremely fast, numerically unstable
`lrs`	`.ext`	LRS	boundary triangulation via LRS; works only if the origin is contained in the polytope interior. computes the exact rational volume.

Table 2: Implemented algorithms

4.3 Further Options

There are additional options which start with a "–" and can be specified at any place in the command line.

The most important feature of `hot`, `rlass` and `rlch` is the ability of storing intermediate results for later use. This behaviour can be controlled via the option `-s` which must be directly (without space) followed by the number of levels for which storing is desired. So `-s0` prevents all storing, and `-s5` allows storing for up to five levels. Of course higher values are preferable, but may exceed the available memory.

The objective functions for `lawd` and `lawnd` are determined randomly. Different objective functions can be obtained by setting a random seed with the option `-r` directly followed by an integer.

The default behaviour when an option is not specified by the user can be controlled using `#define`-sequences in the code; see Section 5.

4.4 Automatic Method Proposal

The detection of a good method follows this scheme: If possible `hot` is chosen, otherwise `rlass` or `lrs` in this order. Note that often `lrs` will be very efficient (and has the additional advantage of computing the exact volume without rounding problems) but is not recommended because the programme cannot completely check its applicability — it does not know if the origin is contained in the polytope interior.

This way of choosing a method has been suggested by extended experiments.

5 Adopting the Code to Your Requirements

Sometimes it will be necessary or useful to change the code according to the type of your volume computation problems. This can be done by editing `vinci.h` and changing the preprocessor definitions at the beginning of the file. Following is a list of the definitions subject to personal changes:

- `LRS_EXEC`: This constant defines how LRS is called. If you prefer to have the executable in a directory different from `vinci-1.0.5`, change the call appropriately by introducing the (absolute or relative) path name.

- `PIVOTING`, `MIN_PIVOT`: The value of PIVOTING determines the strategy for computing determinants (e. g. simplex volumes) by Gaussian elimination for all methods except for `rlass`. If it is 0, the first entry with absolute value bigger than `MIN_PIVOT` is chosen as pivot element; if it is 1, partial pivoting, if it is 2, total pivoting is performed. We obtained results with the maximal machine accuracy using partial pivoting, while the zero value for `PIVOTING` caused numerical problems without speeding up the computations considerably.

- `PIVOTING_LASS`, `MIN_PIVOT_LASS`: The values of these constants are valid for `rlass`. We found that unlike the other algorithms, this method is considerably sped up when using a small `MIN_PIVOT` value.

- `DEFAULT_STORAGE`: The constant is important for methods where intermediate volumes can be stored, i.e. `hot` and `rlass`. It designates the number of recursion levels on which storage is allowed. All these methods are sped up enormously when storing more results. The value is overwritten when the option `-s` is specified.

- `STATISTICS`: If this constant is defined, the programme will output some statistical data. For triangulations and Lawrence's formula this is the total number of simplices (or summed up partial volumes) as well as their distribution into different size classes. The output can be used to detect possible numerical instabilities. For methods storing intermediate results the number of partial volumes computed and retrieved in each dimension is output. If you are not interested in the data replace the definition by `#define NO_STATISTICS`.

- `RATIONAL`: Earlier versions could handle rational numbers; this feature has somehow been lost during the development, but should not be too difficult to implement. If you are interested in exact arithmetics, do not hesitate to contact the authors.

6 Acknowledgements

The first version of VINCI was developed within the ETHZ-EPFL project on Optimisation and Geometric Computation, in close collaboration with Komei Fukuda. We thank Paul A. Vixie for making available to us his efficient

AVL-tree implementation and Jean-Bernard Lasserre for fruitful discussions concerning his volume computation method. We are grateful to Hans-Jakob Lüthi for his kind support and for initiating our project.

References

[1] Benno Büeler, Andreas Enge, and Komei Fukuda. Exact volume computation for polytopes: A practical study. In Gil Kalai and Günter M. Ziegler, editors, *Polytopes — Combinatorics and Computation*, volume 29 of *DMV Seminar*, pages 131–154, Basel, 2000. Birkhäuser Verlag.