Pivot!

Using real algebraic geometry to move a sofa

Corin Lee

Department of Mathematical Sciences
University of Bath

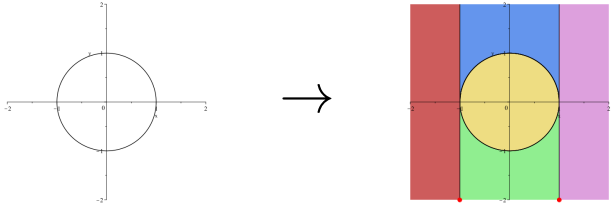
March 9, 2023

Cylindrical Algebraic Decompositions

The aim of this talk is to briefly describe what a CAD is, outline the original CAD algorithm and apply this to some simple examples, then discuss a more real-world example.

A *cylindrical algebraic decomposition* (abbreviated CAD) is a method in real semi-algebraic geometry to decompose $\mathbb{R}^n$ into a finite number of connected semi-algebraic subsets known as *cells*, each homeomorphic to $\mathbb{R}^k$.

CADs can be used to determine if a system of polynomials has a solution by checking the signs of the polynomials in each cell.



Cylindrical Algebraic Decompositions

First arising in a paper by Collins as a sub-algorithm in his work on an effective method for quantifier elimination in real closed fields [Collins, 75], CAD has applications in algebraic simplification technology [Bradford Davenport, 2002] and robot motion planning.

One example of which is the Piano Movers Problem
[Davenport, 1986, Wilson et al, 2013, Schwartz Sharir, 1983]. However, trying to compute even a simple case for this problem can be computationally expensive and impractical.

It is known that the worst-case complexity is doubly-exponential in the number of variables [Davenport Heintz 1987]. Nevertheless, the algorithms are practicable in important cases, and small efficiency gains become important.

There are competing definitions of CAD and different types of CAD, but this talk will cover the original "projection and lifting" method of computing a CAD.

So what is a CAD?

A *cylindrical algebraic decomposition* is in fact a decomposition that is both cylindrical and algebraic.

So what is a CAD?

A *cylindrical algebraic decomposition* is in fact a decomposition that is both cylindrical and algebraic.

But what does *that* mean?

First we have to define (a lot of) things.

Definitions

Let $\mathbb{R}[\mathbf{x}]$ be the polynomial ring over $\mathbb{R}$ with ordered variables $\mathbf{x} = x_1 < \cdots < x_n$.

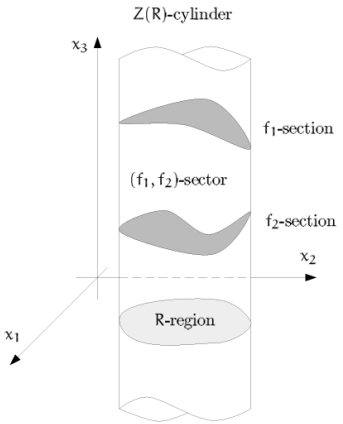
Definition (region, cylinder, section, sector)

For n-dimensional real space $\mathbb{R}^n$,

- **Region,** R**:** a nonempty connected subset of $\mathbb{R}^n$.
- **Cylinder over** R**,** $\mathcal{Z}(R)$**:** the set $R \times \mathbb{R} = \{(\alpha, x) \mid \alpha \in R, x \in \mathbb{R}\}$.

For f, f_1, f_2 continuous, real valued functions of R:

- f**-section of** $\mathcal{Z}(R)$**:** the set $\{\alpha, f(\alpha) \mid \alpha \in R\}$.
- (f_1, f_2)**-sector of** $\mathcal{Z}(R)$**:** the set $\{(\alpha, \beta) \mid \alpha \in R, f_1(\alpha) < \beta < f_2(\alpha)\}$. The functions $f_1 = -\infty$ and $f_2 = +\infty$ are allowed.



Definitions

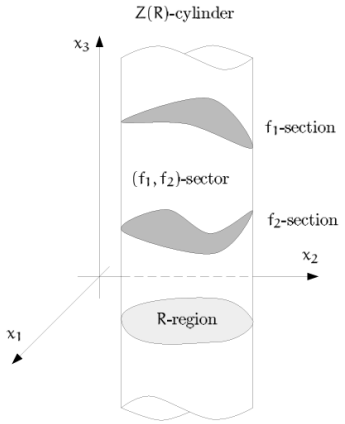
Definition (Decomposition, stack)

A *decomposition* of any $X \subseteq \mathbb{R}^n$ is a finite collection of disjoint regions whose union is X.

Continuous, real-valued functions $f_1 < f_2 < \cdots < f_k$, $k \geq 0$, defined on R, naturally determine a decomposition of $\mathcal{Z}(R)$ consisting of the following regions:

1. the (f_i, f_{i+1})-sectors of $\mathcal{Z}(R)$ for $0 \leq i \leq k$, where $f_0 = -\infty$ and $f_{k+1} = +\infty$, and
2. the f_i-sections of $\mathcal{Z}(R)$ for $1 \leq i \leq k$.

We call such a decomposition a *stack* over R (determined by $f_1, \ldots, f_k$).



Definitions

Definition (Cylindrical decomposition of $\mathbb{R}^n$)

A decomposition $\mathcal{D}$ of $\mathbb{R}^n$ is *cylindrical* if either

1. $n = 1$ and $\mathcal{D}$ is a stack over $\mathbb{R}^0$, or
2. $n > 1$, and there is a unique cylindrical decomposition $\mathcal{D}'$ of $\mathbb{R}^{n-1}$ such that for each region R of $\mathcal{D}'$, some subset of $\mathcal{D}$ is a stack over R.

As $\mathcal{D}'$ is unique, any cylindrical decomposition $\mathcal{D}$ of $\mathbb{R}^n$ will have unique *induced* cylindrical decompositions of $\mathbb{R}^j$ for $j = n-1, n-2, \ldots, 1$. Conversely, given a CAD $\mathcal{D}'$ of $\mathbb{R}^j, j < n$, a CAD $\mathcal{D}$ of $\mathbb{R}^n$ is an *extension* of $\mathcal{D}'$ if $\mathcal{D}$ induces $\mathcal{D}'$.

Alternatively, a decomposition is *cylindrical* if for all $1 \leq j < n$, the *projections* (operations reducing the dimension) on the first j variables of any two regions are either equal or disjoint.

Definitions

Definition (Semi-algebraic set)

A subset of $\mathbb{R}^n$ is *semi-algebraic* if it can be constructed by finitely many applications of the union, intersection, and complementation operations on sets of the form

$$\{x \in \mathbb{R}^n \mid f(x) = 0\} \text{ or } \{x \in \mathbb{R}^n \mid f(x) > 0\}$$

where $f \in \mathbb{R}[\mathbf{x}]$.

Definition (Algebraic decomposition)

A decomposition is *algebraic* if each of its regions is a semi-algebraic set.

and finally...after five pages of definitions:

Definition (Cylindrical Algebraic Decomposition)

A *cylindrical algebraic decomposition*, or *CAD*, of $\mathbb{R}^n$ is a decomposition which is both cylindrical and algebraic:

- **Decomposition:** A *decomposition* of $X \subseteq \mathbb{R}^n$ is a finite collection of disjoint cells whose union is X.
- **Cylindrical:** A decomposition is *cylindrical* if for all $1 \leq j < n$, the projections on the first j variables of any two cells are either equal or disjoint.
- **Algebraic:** A decomposition is *algebraic* if each of its regions is a set that can be constructed by finitely many unions, intersections and complementations on sets of the form $\{x \in \mathbb{R}^n \mid f(x) = 0\}$ or $\{x \in \mathbb{R}^n \mid f(x) > 0\}$, where $f \in \mathbb{R}[\mathbf{x}]$.

The components of a CAD are called *cells*, and for $0 \leq j \leq n$, a *j-cell* in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ which is homeomorphic to $\mathbb{R}^j$.

Definitions

We often want the decomposition to respect some collection of polynomials:

Definition ($\mathcal{F}$-invariant)

Let $\mathcal{F} = \{f_i \in \mathbb{R}[x_1, \ldots, x_n], 1 \leq i \leq r\}$ and $X \subseteq \mathbb{R}^n$. We say X is *$\mathcal{F}$-invariant* (and $\mathcal{F}$ is *invariant* on X) if each $f_i(x)$ has constant sign for every $x \in X$, that is,

$$\forall x \in X : f_i(x) \diamond 0,$$

where $\diamond \in \{>, =, <\}$. A polynomial $f_i(x)$ with constant sign in this sense is called *sign-invariant*.

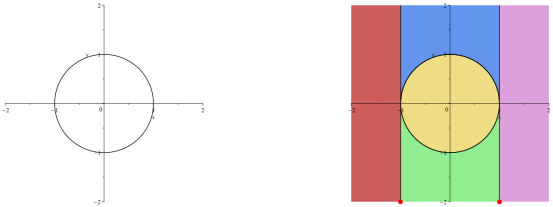


Figure: The graph of $x^2 + y^2 - 1$ and the $\{x^2 + y^2 - 1\}$-invariant CAD of $\mathbb{R}^2$.

Approaches and Implementations

Now we know what a CAD is, how do we implement it?

The original implementation by Collins of an algorithm for producing a CAD does so by taking a set of polynomials $\mathcal{F} \subset \mathbb{R}[\mathbf{x}]$ and produces an $\mathcal{F}$-invariant CAD of $\mathbb{R}^n$. This type of algorithm is known as a *projection and lifting* CAD algorithm, or PL-CAD:

The set of polynomials are "projected" down into fewer and fewer variables until a set of univariate polynomials are reached. The zero set of these polynomials and the intervals between them form a stack, and taking sample points in each sector and section forms a CAD of $\mathbb{R}^1$. The CAD is now "lifted" to one of $\mathbb{R}^2$ by substituting these sample points into the bivariate projection polynomials and repeating the process, and this process is repeated until a CAD of $\mathbb{R}^n$ is produced.

There have been implementations of PL-CAD in various software, along with variants and improvements, which usually aim to produce better performance under certain conditions.

PL-CAD

The PL-CAD algorithm can be split into three phases:

- **Projection:** Repeatedly apply a projection operator *Proj*:

$$\mathcal{F} = \mathcal{F}_n(x_1, \ldots, x_n) \xrightarrow{Proj} \mathcal{F}_{n-1}(x_1, \ldots, x_{n-1}) \xrightarrow{Proj} \ldots \xrightarrow{Proj} \mathcal{F}_1(x_1)$$

  The zero sets of the polynomials produced by each step are the projections of "significant points" of the previous set of polynomials.

- **Base phase:** Real root isolation on $\mathcal{F}_1(x_1)$, roots and open intervals between form an $\mathcal{F}_1$-invariant CAD of $\mathbb{R}^1$.
- **Lifting phase:** For each cell C of the $\mathcal{F}_{k-1}$-invariant CAD in $\mathbb{R}^{k-1}$, construct a sample point s and isolate the real roots of the polynomials of $\mathcal{F}_k$ at s. The sectors and sections of these polynomials form a stack, and these stacks make the cells of the $\mathcal{F}_k$-invariant CAD of $\mathbb{R}^k$ above C.

It is enough to determine the signs of $\mathcal{F}$ in these sample points as each cell is $\mathcal{F}$-invariant by construction.

An Example in Three Dimensions

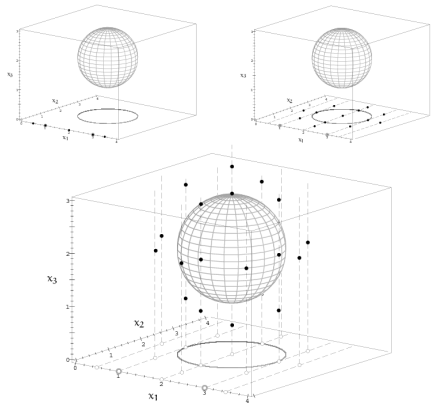


Figure: Sample points for the unit sphere centred at $(2, 2, 2)$, $f = (x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 2)^3 - 1$.

The Projection Operator

So what is this projection operator, what are these "significant points" and how are they calculated?

In short, the set of projection polynomials of $\mathcal{F}$ involve points such as vertical tangents of each polynomials and crossings between them. Fully describing the projection operator would involve defining another dozen things (to the point where many papers omit it!), but we will focus on some key components.

In a simple sense, for $\mathbf{x} = x_1 < \cdots < x_n$ and $\mathcal{F} = \{f_i \in \mathbb{R}[\mathbf{x}], 1 \leq i \leq r\} \subseteq \mathbb{R}[\mathbf{x}]$, Proj takes each polynomial $f_i \in \mathcal{F}$ and treats it as a univariate polynomial in its greatest variable, i.e. $f_i \in \mathbb{R}[x_1, \ldots, x_{n-1}][x_n]$. It then looks at the coefficients of each polynomial (in $\mathbb{R}[x_1, \ldots, x_{n-1}]$), as well as their *resultants* and *discriminants*.

The Sylvester Matrix

Let $f = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$ and $g = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0$. The *Sylvester matrix*, $\mathrm{Syl}(f, g)$ is the $(m+n) \times (m+n)$ matrix defined as follows:

$$\mathrm{Syl}(f,g) = \underbrace{\begin{pmatrix} a_m & a_{m-1} & \cdots & a_0 & 0 & \cdots & \cdots & 0 \\ 0 & a_m & a_{m-1} & \cdots & a_0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & a_m & a_{m-1} & \cdots & a_0 \\ b_n & b_{n-1} & \cdots & b_0 & 0 & \cdots & \cdots & 0 \\ 0 & b_n & b_{n-1} & \cdots & b_0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & b_n & b_{n-1} & \cdots & b_0 \end{pmatrix}}_{m+n} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} n \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} m \end{matrix}$$

The *resultant* of f and g, denoted $\mathrm{res}(f, g)$ is the determinant of $\mathrm{Syl}(f, g)$. The *discriminant* of f, denoted $\mathrm{disc}(f)$, is $\mathrm{res}\left(f, \frac{\mathrm{d}f}{\mathrm{d}x}\right)$.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$. You can tell I'm proud of the pictures. Recall this is treated as a polynomial in y.

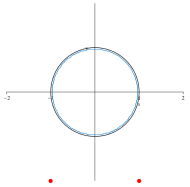


Figure: This bad boy, now with dots!

The Projection Phase

- Coefficients: $\{1, x^2 - 1\}$.
- Resultants: None!
- Discriminants: $disc(f) = \begin{vmatrix} 1 & 0 & x^2-1 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{vmatrix} = 4(x^2 - 1)$.

The only real zeroes of these polynomials are $x = \pm 1$, which correspond to the vertical tangents.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$.

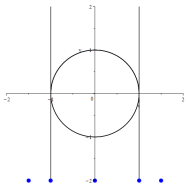


Figure: Now with more dots, and lines!

The Base Phase

We now have the stack over $\mathbb{R}^1$: $(-\infty, -1), -1, (-1, 1), 1, (1, \infty)$ and thus a CAD with those cells.

We can then take some "nice" sample points. In the picture we can see that the x-axis is now decomposed into five cells with with sample points $-1.5, -1, 0, 1, 1.5$.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$.

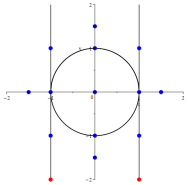


Figure: We pick sample points for each cell of $\mathbb{R}^2$.

The Lifting Phase

Substitute these sample points into f:

$x = -1.5:$	$f = y^2 + 1.25$	$+$ always	(1 region)
$x = -1:$	$f = y^2$	$+$ when $y > 0$, $y < 0$	
		0 when $y = 0$,	(3 regions)
$x = 0:$	$f = y^2 - 1$	$+$ when $y > 1$, $y < -1$,	
		0 when $y = 1$, $y = -1$,	
		$-$ when $-1 < y < 1$,	(5 regions)
$x = 1:$	$f = y^2$	$+$ when $y > 0$, $y < 0$	
		0 when $y = 0$,	(3 regions)
$x = 1.5:$	$f = y^2 + 1.25$	$+$ always	(1 region)

That means there are 13 regions in total. If we choose some nice sample points, we're done!

A very simple example

We can see that even for such a simple example, we have decomposed $\mathbb{R}^2$ into 13 cells!

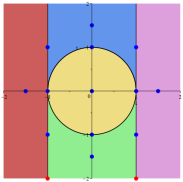


Figure: The f-invariant CAD of $\mathbb{R}^2$.

Using *Maple*, we can see descriptions of all cells:

$$
cad := \left\{
\begin{array}{ll}
1 & x < -1 \\
\left\{ \begin{array}{ll} 1 & y < 0 \\ 1 & y = 0 \\ 1 & 0 < y \end{array} \right. & x = -1 \\
\left\{ \begin{array}{ll} 1 & y < -\sqrt{-x^2+1} \\ 1 & y = -\sqrt{-x^2+1} \\ 1 & -\sqrt{-x^2+1} < y \wedge y < \sqrt{-x^2+1} \\ 1 & y = \sqrt{-x^2+1} \\ 1 & \sqrt{-x^2+1} < y \end{array} \right. & -1 < x \wedge x < 1 \\
\left\{ \begin{array}{ll} 1 & y < 0 \\ 1 & y = 0 \\ 1 & 0 < y \end{array} \right. & x = 1 \\
1 & 1 < x
\end{array}
\right.
$$

A less simple example

For a slightly more complicated set of polynomials, it's even more complicated! With two polynomials in two dimensions, we end up with 51 cells!:

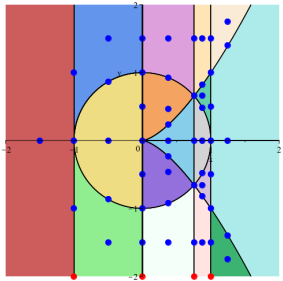


Figure: The $\mathcal{F}$-invariant CAD of $\mathbb{R}^2$ for $\mathcal{F} = \{x^2 + y^2 - 1, x^3 - y^2\}$.

(Note the points on $\mathbb{R}^1$, -1 and $+1$ correspond to the discriminant of $x^2 + y^2 - 1$, 0 corresponds to the discriminant of $x^3 - y^2$ and $\alpha \approx 0.7549$ corresponds to the resultant of the two polynomials.)

"That's great, but what about my sofa?"

We've seen some examples of CADs, but how about something more real-world? Consider the Piano Movers Problem, a robotics problem defined by [Schwartz Sharir, 1983] as follows:

"Given a body B and a region bounded by a collection of walls,

"That's great, but what about my sofa?"

We've seen some examples of CADs, but how about something more real-world? Consider the Piano Movers Problem, a robotics problem defined by [Schwartz Sharir, 1983] as follows:

"Given a body B and a region bounded by a collection of walls, either find a continuous motion connecting two given positions and orientations of B during which B avoids collisions with the walls,

"That's great, but what about my sofa?"

We've seen some examples of CADs, but how about something more real-world? Consider the Piano Movers Problem, a robotics problem defined by [Schwartz Sharir, 1983] as follows:

"Given a body B and a region bounded by a collection of walls, either find a continuous motion connecting two given positions and orientations of B during which B avoids collisions with the walls, or else establish that no such motion exists."

Modelling the Piano Movers Problem

Moving a piano is a tricky task in the real world, and modelling it will be more complicated than the previous examples.

How can we model it effectively, and how will CAD deal with it?

Quantifier Elimination

A *Tarski formula* is a formula, made up of equations, inequations and inequalities, as well as the quantifiers $\exists$ and $\forall$ and Boolean connectives.

Quantifier elimination is the concept of taking a quantified statement and returning an equivalent statement without quantifiers. For example, the statement

$$\exists x \in \mathbb{R}(a \neq 0 \wedge ax^2 + bx + c = 0)$$

can be made into

$$a \neq 0 \wedge b^2 - 4ac \geq 0.$$

Modelling the Piano Movers Problem

[Davenport, 1986] gives a simple example of the problem of moving a ladder of length 3 through a right-angled corridor of width 1, and considered building a CAD to solve the problem.

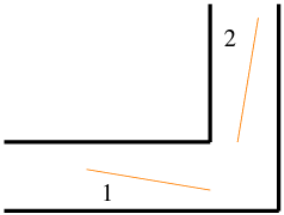


Figure: Can the ladder move from position 1 to position 2?

Modelling the Piano Movers Problem

[Davenport, 1986] gives a simple example of the problem of moving a ladder of length 3 through a right-angled corridor of width 1, and considered building a CAD to solve the problem.

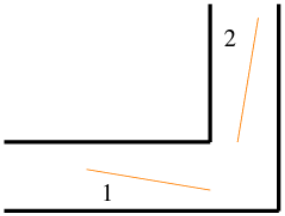
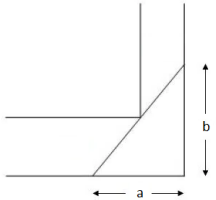


Figure: Can the ladder move from position 1 to position 2?
Show of hands!

Modelling the Piano Movers Problem

We can actually solve this relatively easily: the longest possible ladder that could fit would touch both walls and the inside corner.



By similar triangles, we have $\frac{a-1}{1} = \frac{1}{b-1}$, leading to the maximisation problem:

$$\max a^2 + b^2 \text{ subject to } a \leq \frac{1}{b-1} + 1$$

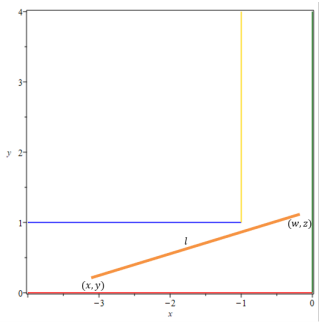
of which the solution is $\sqrt{8}$. So this ladder won't fit!

Modelling the Piano Movers Problem

How do we model this for a CAD? Denoting the endpoints of the ladder as (x, y) and (w, z) and assuming the outer corner of the corridor is the origin, we have a ladder given by $(x-w)^2+(y-z)^2=9$ and walls given by $\{y=0, x<0\}$, $\{y=1, x<-1\}$, $\{y<0, x=0\}$, $\{y>1, x=-1\}$.

The formulation of this is:

$$
\begin{aligned}
&\big[(x-w)^2+(y-z)^2-9=0\big]\\
\wedge\ &\big[[yz\geq 0]\vee[x(y-z)^2+y(w-x)(y-z)\geq 0]\big]\\
\wedge\ &\big[[(y-1)(z-1)\geq 0]\\
&\quad\vee[(x+1)(y-z)^2+(y-1)(w-x)(y-z)\geq 0]\big]\\
\wedge\ &\big[[xw\geq 0]\vee\ [y(x-w)^2+x(z-y)(x-w)\geq 0]\big]\\
\wedge\ &\big[[(x+1)(w+1)\geq 0]\\
&\quad\vee[(y-1)(x-w)^2+(x+1)(z-y)(x-w)\geq 0]\big].
\end{aligned}
$$



The first equality describes the ladder and last 8 inequalities describe valid positions such that the ladder does not intersect with the four walls (the two ends are on the same side of the wall, or the ladder is on the other side of the line touching the end of the wall).

What does the CAD look like?

This is a CAD problem over $\mathbb{R}^4$ involving 9 polynomials, already much bigger than the previous examples!

In fact, even after simplifying these polynomials, the projection into three dimensions gives us 32 polynomials, the projection into two dimensions gives us 79 polynomials. These can be reduced to 24 distinct polynomials, the largest of which is of total degree six. The final projection gives 250 distinct polynomials, the highest degree of which is 26.

computing the construction of the CAD from this formulation was not possible at the time and is still not computationally feasible with today's hardware.

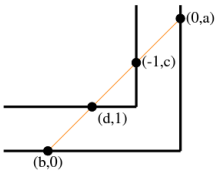
So, we don't know!

In robotics, Piano Movers Problems would typically be tackled using numerical methods to produce paths efficiently at the expense of the possibility of rounding errors, however we are interested in a symbolic approach. There are however other formulations which are more promising.

Alternate formulations

A reformulation by Wang notes that the ladder cannot traverse the corridor if and only if it intersects all four walls simultaneously: Let a, b, c, d be coordinates defining the intersection points as below and ℓ be the length of the ladder). Then there is no solution if

$$(\exists a)(\exists b)(\exists c)(\exists d)$$
$$[a^2 + b^2 = \ell^2 \wedge \ell > 0 \wedge a \geq 0 \wedge b < 0 \wedge c \geq 1 \wedge d < -1$$
$$\wedge c - (1 + b)(c - a) = 0 \wedge d(1 - a)(d - b) = 0].$$



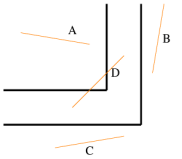
This makes a huge difference, allowing the program QEPCAD to deduce the maximal ladder length is $\sqrt{8}$ using a CAD of 19 cells. However, this requires geometric deductions before the problem is presented to CAD, and informs you only whether the ladder can or cannot pass through the corridor.

Alternate formulations

[Wilson et al, 2013] gives a new formulation describing invalid configurations.

Characterising the invalid regions with these, eliminating the variable t and negating this gives the valid regions. Adding to this the formula fixing the length of the ladder, we have:

A	$x < -1 \wedge y > 1$ or $w < -1 \wedge z > 1$: this describes any collision with the 'inside' walls along with the ladder being on the other side of these.
B	$x > 0$ or $w > 0$: this describes any collision with the rightmost wall along with the ladder being on the other side.
C	$y < 0$ or $z < 0$: this describes any collision with the bottommost wall along with the ladder being on the other side.
D	$(\exists t)[0 < t \wedge t < 1 \wedge x + t(w - x) < -1 \wedge y + t(z - y) > 1]$: this ensures no inner point of the ladder lies in the invalid top-left region.



$$
\begin{aligned}
&[(x-w)^2 + (y-z)^2 = 9] \wedge \\
&[w \leq 0] \wedge [x \leq 0] \wedge [y \geq 0] \wedge [z \geq 0] \wedge [x \geq -1 \vee y \leq 1] \\
&\wedge [w \geq -1 \vee z \leq 1] \wedge \big[wy - w + x + y < 0 \vee w + 1 \geq 0 \\
&\quad \vee xz + z - yw + w - y - x \leq 0\big] \\
&\wedge \big[yw - w + y + x \geq 0 \vee \big[[z - 1 \leq 0 \\
&\quad \vee xz + z - yw + w - y - x \geq 0] \wedge y - 1 \leq 0\big]\big].
\end{aligned}
$$

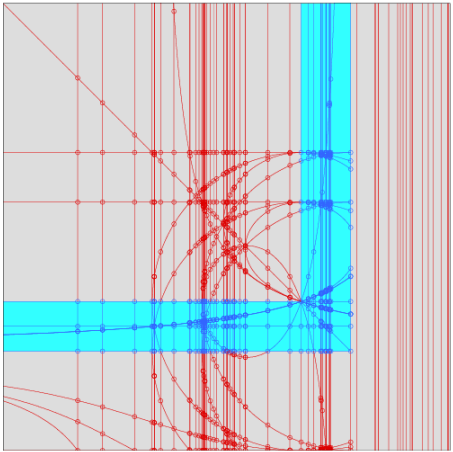
Giving this to QEPCAD with the variable ordering $x < y < w < z$ took just under 5 hours to construct a CAD of $\mathbb{R}^4$ with over 285,000 cells, however introducing quantifiers on one endpoint dramatically speeds this up.

Alternate formulations

Simply adding $(\exists w)(\exists z)$ to the start of the previous formulation reduces the time to around 50 minutes and the CAD to 5,453 cells, resulting in the quantifier-free formula

$$x \leq 0 \wedge y \geq 0 \wedge [x + 1 \geq 0 \vee y - 1 \leq 0],$$

which just describes the original corridor, as this is just asking for the points where it is possible to place an end of the ladder have it in a valid position. This has dramatically reduced the complexity of the problem and while it is not sufficient to find a path through when possible, it can be useful to test if such a path exists. Despite this, a depiction of the two-dimensional CAD of the (x, y) configuration space makes it clear just how complicated the problem is when tackled by CAD:



So in short, you certainly could try to use CADs to see if you are able to move your sofa through a corridor, but it would probably be easier and far, far quicker to use another method, or simply try and give it a go.
CADs are a powerful tool for solving polynomial systems and quantifier elimination, but their inherent complexity means they can quickly become infeasible. However, this also means even the smallest gains from alternate methods can have huge benefits, and as improvements continue to be made and computing power increases, the scope of the problems that can be solved with CADs is always increasing.

References


Collins, George (1975)
Quantifier elimination for real closed fields by cylindrical algebraic decomposition
Lecture Notes in Computer Science.


Arnon, Dennis and Collins, George and McCallum, Scott (1984)
Cylindrical Algebraic Decomposition I: The Basic Algorithm
SIAM J. Comput. 13(11), 865 – 877.


Jirstrand, Mats (1995)
Cylindrical Algebraic Decomposition - an Introduction


Chen, Changbo and Moreno Maza, Marc (2009)
Computing cylindrical algebraic decomposition via triangular decomposition
Proceedings of the 2009 international symposium on Symbolic and algebraic computation, 95 – 102.


Chen, Changbo and Moreno Maza, Marc (2012)
An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions


Davenport, James (1986)
A "piano movers" problem

References


Bradford, Russell and Davenport, James H. (2002)

Towards Better Simplification of Elementary Functions


Schwartz, Jacob T. and Sharir, Micha (1983)

On the "piano movers" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers


Wilson, David and Davenport, James H. and England, Matthew and Bradford, Russell (2013)

A "Piano Movers" Problem Reformulated


Davenport, J. H. and Locatelli, A. F. and Sankaran, G. K. (2019)

Regular cylindrical algebraic decomposition


Davenport, James H. and Heintz, Joos (1987)

Real quantifier elimination is doubly exponential


Chen, Changbo and Moreno Maza, Marc

The RegularChains Library `https://www.regularchains.org/`