International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 149-157, June-2025

https://doi.org/10.47001/IRJIET/2025.906019

Enhancing AutoCAD with the Ability to Create Bidirectional Equality and Inequality Constraints on Line Segment Lengths

Pham Tuan Anh

School of Mechanical Engineering, Hanoi University of Science and Technology (HUST), Vietnam
E-mail: anh.phamtuan@hust.edu.vn

***Abstract* - In CAD systems, the integration of parametric functionalities can provide practical benefits. This is evident in AutoCAD.**

In this paper, a bidirectional constraint implemented in AutoCAD is understood as a constraint between variables such that when any one variable changes, the other variables involved are automatically adjusted by AutoCAD to maintain the constraint.

Based on the rich set of equality and inequality relations in planar geometry, it is possible to construct a library of base blocks. Each base block contains a geometric figure that realizes a specific constraint among the lengths of certain Line objects. This paper presents a method for supplementing AutoCAD with commands to coordinate the base functions, which automatically insert the necessary base blocks, decompose the resulting objects, and add supplementary constraints. As a result, it is possible to ensure the desired bidirectional constraints in the form of equalities or inequalities of the lengths of Line objects in the drawing.

Additionally, there are commands to manage the created constraints, such as displaying the symbolic form of the constraint, identifying the participating objects, and removing constraints. The described method is also useful when a complex constraint can be replaced by a system of constraints available in AutoCAD, along with the types of constraints introduced here.

Keywords: AutoCAD, parametric constraint, geometric constraint, dimension constraint, VBA macro, bidirectional constraint, unidirectional constraint.

I. GENERAL INTRODUCTION

AutoCAD is widely used CAD software across many fields. AutoCAD is equipped with parametric functionalities at a certain level and is capable of creating and managing constraints directly in two-dimensional space. Solvers for 2D and 3D geometric constraints are also integrated into many other CAD applications. Over time, numerous studies have focused on improving constraint-solving methods, expanding the types of geometric objects and constraints that a CAD system can recognize and process ([1]), refining the classification and selection of constraints to enable reusable models that can generate various geometric variants while preserving the essential design intent ([2]). An effective approach is to leverage and extend certain results from elementary geometry to replace complex constraints with equivalent systems of simpler constraints.

In AutoCAD, it is possible to apply several types of geometric constraints ([3, 4]) such as "Equal", "Coincident", "Parallel", and other available dimension constraints (the term used in AutoCAD). However, once a "dimension constraint" tool is used to force a length y_1 to be expressed as a specified function of y_2, y_3 (in essence, assigning an expression involving parameters), it is generally not possible to arbitrarily choose a new value for y_1 and have AutoCAD automatically solve the inverse problem to find and assign new values for y_2 and y_3 that satisfy the predefined functional relationship.

There are also situations requiring the imposition of inequality constraints on lengths to define the allowable range for certain objects, such that when editing shapes using the STRETCH command, the constraint is still maintained. In general, this capability is not richly available for direct use in AutoCAD.

With a constraint solver, suppose there exists a constraint among several variables, and when any variable changes, the other participating variables are automatically recalculated to preserve the constraint. In this paper, such a constraint is referred to as a bidirectional constraint. A constraint that is not bidirectional is referred to as a unidirectional constraint. Section II will propose a method for creating a type of bidirectional constraint in the form of equalities or inequalities of the lengths of line segments to further enhance the capability of establishing bidirectional constraints in AutoCAD.

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 149-157, June-2025
https://doi.org/10.47001/IRJIET/2025.906019

II. CORE PRINCIPLES OF THE CONSTRAINT CREATION METHOD

In planar geometry, suppose we construct a suitable figure Φ, hereafter referred to as a base figure, whose properties guarantee a certain relation Ω in the form of an equality or inequality involving the lengths of several line segments. If additional "Equal" geometric constraints (in AutoCAD) can be imposed between the suitable line segments belonging to Φ and other line segments in the drawing, then a new relation $\tilde{\Omega}$ among the lengths of suitable line segments outside of Φ can be established, where $\tilde{\Omega}$ has the same form as Ω. When editing objects in the drawing, AutoCAD will automatically seek a geometric solution that satisfies the relation $\tilde{\Omega}$ if such a solution exists. Naturally, finding a solution also depends on the solver's capability. If y_1 is one of the variables participating in $\tilde{\Omega}$, then, in general, when y_1 is modified, the remaining variables will be automatically adjusted so that $\tilde{\Omega}$ remains satisfied. In this way, a bidirectional constraint is created. This capability is extremely useful for the geometric design process, as it enables the reshaping of sketches and the generation of multiple design alternatives for evaluation and selection.

In each base figure, every line segment is represented by a "Line object" in AutoCAD. A library of base figures can be established, and functions can be programmed to facilitate copying a figure from the library and assigning the corresponding constraint, as defined by that planar figure, to other Line objects in the drawing. Each function can return the handle of an object as a result. Constraints created in this manner are referred to as base constraints, and the corresponding functions are called base functions. Nested function calls can be created from these base functions, such that the result returned by one function serves as an argument for another, thereby enabling the creation of even more complex constraints from the base constraints.

If the function call is written in the syntax of the AutoLISP language (see Section V), then when this call is entered on the AutoCAD command line, AutoCAD will parse and execute the function call accordingly.

The base figures are drawn in advance. From these figures, blocks are created. These blocks are called base blocks and are stored in separate files. These files are placed in a default directory. The BEDIT command in AutoCAD can be used to edit these blocks, allowing the user to check, modify, and supplement the constraints present within each block.

The following sections will elaborate on the following aspects:

- A summary of the program's operation when creating constraints.
- The general principles and algorithms of the commands, functions, and procedures used to input and create constraints.
- The structure of base blocks in the library.
- Examples of specific base blocks.
- Commands for managing constraints established in the drawing.

III. EXAMPLE OF A CONSTRAINT SYSTEM ASSIGNED TO A PLANAR FIGURE

Consider the sketch of the front view of a solid of revolution, where the volume of the lower part of the solid has already been determined. The dimensions d and h are also restricted by specific value ranges. The system of constraints is given as follows:

$$\begin{cases} 20 \le \mathrm{d} \le 30 \\ 25 \le \mathrm{h} \le 40 \\ \mathrm{a.b}^2 = 19099 \end{cases}$$

After these constraints are assigned, it is possible to resize the figure using the STRETCH command and to assign specific values to the dimensions to obtain different design alternatives, as shown in Figure 1, parts (a) and (b).

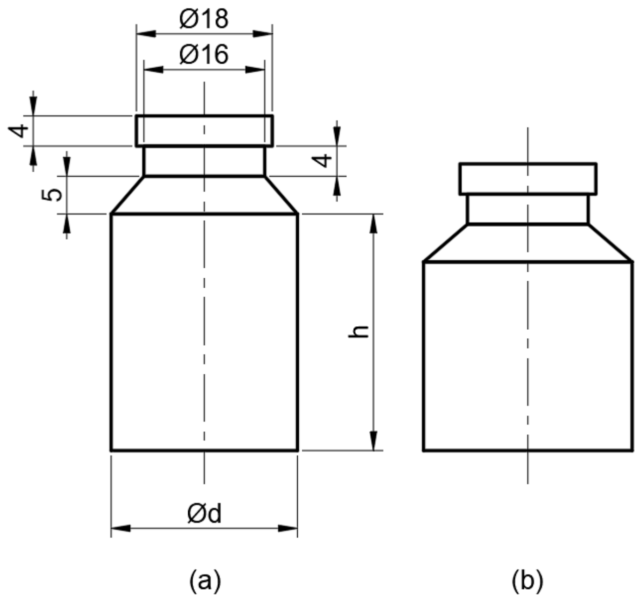

Figure 1: Two different design alternatives of the front view of the solid

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 149-157, June-2025

https://doi.org/10.47001/IRJIET/2025.906019

IV. SUMMARY OF THE WORKFLOW FOR THE PROGRAM AND USER IN CREATING A CONSTRAINT IN THE DRAWING

4.1 The user enters the INPC command.

4.2 The program prompts the user to enter a description string for the constraint and select the objects corresponding to the variables in that constraint.

4.3 The program creates a group to contain the selected objects, and later to include additional objects originating from blocks retrieved from the library to construct the constraint.

4.4 The program converts the description string into a base function call or nested base function calls following the syntax of the AutoLISP language, then enters the function call on the AutoCAD command line for execution.

4.5 When executed, each base function inserts a block from the library into the drawing as a block reference, explodes the block reference into a set of objects, and creates supplementary constraints between the objects in this set and the objects initially selected by the user in step 4.1 to implement the constraint.

Note: Steps 4.2, 4.3, and 4.4 are performed by the INPC command (see Section V). Step 4.5 is performed by the base functions (see Section VIII).

V. THE INPC COMMAND

5.1 Introduction

The INPC command can be entered at the AutoCAD command line. This command executes the procedure Inpc_, which is written in VBA (Visual Basic for Applications) ([5]). The program will prompt the user to enter a character string $\mathcal{S}$, which serves as the constraint description, and it will call the base functions to assign the constraints.

5.2 Detailed Description of the INPC Command

The description string is written in parenthesis syntax, similar to an expression in the AutoLISP language, but must be entered on a single line. In the simple case, the format is as follows:

"(<Base_constraint_name> <Argument_1> <Argument_2> …)"

In complex cases, an argument can itself be replaced by another description string; that is, parentheses can be nested. In this case, the elements in the string are separated by spaces or tabs and by parentheses.

The name of the base constraint is actually the abbreviation for a base function, such as "+", "-", "*/", "<=", etc. (see Section X). In some situations, an argument may be nil if it is not required (see Section 6.1). If an argument is a numeric constant, it must conform to the Double or Long data types in VBA ([5]). Apart from these cases, the remaining arguments are named p1, p2, p3, ... in left-to-right order within the description string.

Examples:

- The string "(*/ p1 p2 p3 p4)" corresponds to the constraint $p_1 = (p_2 \cdot p_3)/p_4$
- The string "(*/ p1 10 p2 (+ nil p3 p4))" corresponds to the constraint $p_1 = (10 \cdot p_2)/(p_3 + p_4)$
- The string "(<= p1 (<= p2 p3))" corresponds to the constraint $p_1 \le p_2 \le p_3$

(The base constraint named "≤" is described in Section X.)

In the description string, the substrings "p1", "p2", "p3", etc. correspond to the objects $\overline{p}_1$, $\overline{p}_2$, $\overline{p}_3$, etc., which are the Line objects to be input, having lengths p_1, p_2, p_3, etc., respectively.

The program will sequentially display N prompts to select the Line objects $\overline{p}_i$ $(i = \overline{1, N})$.

The program will find the handle of each selected object and create an array $\mathcal{H}$ to store these handles.

The program then creates a group G containing the selected objects. The group name represents a positive integer, such as 1, 2, …, and must not duplicate any existing group name in the drawing.

Each group in AutoCAD has an extension dictionary that stores additional information. The constraint description string $\mathcal{S}$ will be written to an xrecord in this extension dictionary with the key "Description". The elements of array $\mathcal{H}$ are also written to an xrecord with the key "Handles". These xrecords are necessary when the user needs to find information about the constraint and the participating objects. If a drawing object participates in multiple constraints, it will simultaneously belong to multiple groups.

From the string $\mathcal{S}$, the program constructs a new string $\mathcal{S}'$ in the form of an AutoLISP expression by applying the following transformations:

- Replace the names of the base constraints with the names of predefined base functions (e.g., "+" is replaced by "Addi"; see Section 10.3).

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 149-157, June-2025
https://doi.org/10.47001/IRJIET/2025.906019

- Append the name of the group G (e.g., "1") after the base function name.
- Replace the substrings "p1", "p2", ... with the handles of the corresponding objects $\overline{p}_1$, $\overline{p}_2$, ...
- Add a space at the end of the string.

Example:

If $\boldsymbol{S}$ is "(<= p1 (<= p2 p3))" then $\boldsymbol{S}'$ could be "(Leoe "1" "2C3" (Leoe "1" "2C6" "28C"))" for a given suitable drawing.

The program will send the string $\boldsymbol{S}'$ to the AutoCAD command line to execute the base function or the nested base function calls. The result is stored in the AutoCAD system variable USERS5 as a string, from which pairs of the form (h_a, h_x) (1) or (h_y, λ) (2) can be inferred. Here, h_a, h_x, h_y are the handles of the objects $\overline{a}$, $\overline{x}$, $\overline{y}$, and λ is a numeric constant.

The program calls the Asco procedure to read the USERS5 variable and to assign the constraints $a = x$ for pairs of type (1) and $y = \lambda$ for pairs of type (2), where a, x, and y are the lengths of the Line objects $\overline{a}$, $\overline{x}$, $\overline{y}$, respectively. These names and notational conventions will continue to be used in Section VI.

VI. BASE FUNCTION FOR ASSIGNING BASE CONSTRAINTS

6.1 General Introduction

This function is implemented in the AutoLISP language. The base function can be called by entering it on the AutoCAD command line. This input is performed automatically during the execution of the INPC command (see Section V).

Syntax of the base function call:

(f Gname h_1 h_2 h_3 …)

- f is the function name, reminiscent of the constraint being created, e.g., Addi, Subtr, etc.
- Gname is the name of the group that was created by the INPC command to contain the objects to be assigned constraints and the figures originating from the base block library.
- h_1, h_2, h_3, … can be numeric constants or the handles of objects to which constraints will be assigned (see Section 5). In this context, “object” refers specifically to a Line entity in an AutoCAD drawing, each of which possesses a unique handle.
- h_1 can also be nil.

The return value of the base function:

If h_1 is the handle h_a of an original object $\overline{a}$ in the drawing, the base function will assign the constraint $a = x$, where $\overline{x}$ is a suitable object from the base figure. The base function returns h_x, the handle of $\overline{x}$. In the case of nested function calls, h_x can be used as an argument for another base function.

If the only purpose is to generate an argument for another function without assigning a base constraint to any specific object in the original drawing, simply replace h_1 with nil in the function call. For example, if you only need to create a Line object whose length equals $b + c$ to participate in another constraint, the call to the Addi function must follow the template below (see also Section 10.3):

(Addi Gname nil h_b h_c)

Here, the Line objects $\overline{b}$ and $\overline{c}$ have lengths b and c, respectively, and their handles are h_b and h_c, respectively.

That call is placed inside another function call, for example:

(Leoe "1" "278" (Addi "1" nil "274" "275"))

6.2 Tasks Performed by the Base Function

6.2.1 Inserting the Base Block Retrieved from the Library

The name of the file that contains the block (excluding the file extension) must match the function name f. The result will be a block reference β. The program prompts for the insertion point of β. When inserting, the rotation angle is set to 0, and the X and Y scale factors are set to 1.

After insertion, β is exploded using the AutoCAD EXPLODE command, generating objects that form the base figure. The objects of the base figure are collected into the list χ and are also added to the group named Gname.

With AutoCAD 2025, after such an explosion, some constraints present in the block may not be maintained, as they are no longer compatible (as per AutoCAD regulations). According to the instructions in [4], the ability and manner of retaining dimensional constraints also depend on the value of the system variable PARAMETERCOPYMODE. When creating figures for constructing base blocks, it is necessary to experiment and anticipate which constraints will be discarded.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 149-157, June-2025
https://doi.org/10.47001/IRJIET/2025.906019

6.2.2 Creating the List of Supplementary Constraints to be Assigned to Objects

- These constraints are of the forms a = x and y = λ.
- **Type a = x:**
- The Line object $\bar{a}$, with handle h_a, is among h_1, h_2, h_3, ... (the arguments of function *f*).
- $\bar{x}$ is a suitable Line object from the base figure just created (see Section 6.2.1).
- **Type y = λ:**
- $\bar{y}$ is a suitable Line object from the base figure, and λ is a numeric constant.
- Accordingly, the supplementary constraints correspond to the pairs (h_a, h_x) and (h_y, λ). Each such pair is converted into two separate strings, which are then concatenated and appended to the content of the AutoCAD system variable USERS5, forming an extended string. In the resulting string, a space is inserted between every two consecutive components that are appended. All numeric constants are stored as strings that represent floating-point numbers (i.e., real numbers with a decimal point), so each string contains a decimal point, making it easy to distinguish them from handles.

Example content of the USERS5 system variable:
"274 295 296 100.0 277 2AE 274 2AD"

6.2.3 Base Function Return Value

The result returned is the handle of the object $\bar{x}$, which is appropriately selected from the base figure, according to the principle described at the end of Section 6.1.

VII. ASCO PROCEDURE FOR ASSIGNING CONSTRAINTS BASED ON AN EXISTING LIST

- This procedure is called from the Inpc_ procedure.
- It is written in VBA and integrates an AutoLISP function, which is used to perform a specialized task. In addition, to simplify and speed up the constraint assignment process, it is possible to create a supplementary command written in VB.NET to take advantage of the AutoCAD .NET API.
- The tasks of the Asco procedure are as follows:
  - Read the string stored in the USERS5 system variable. Recognize whitespace characters as delimiters to split the string into its components. Select consecutive pairs of components, then process them to obtain pairs of the forms (h_a, h_x) (1) and (h_y, λ) (2) (see Section 6.3).
  - Assign the “Equal” geometric constraint for pairs of type (1) and the “Aligned” dimension constraint (in AutoCAD) for pairs of type (2).

VIII. COMMANDS FOR MANAGING CONSTRAINTS IN THE DRAWING

These commands are additionally defined and can be entered on the AutoCAD command line. Each command calls a pre-defined VBA macro.

8.1 The GSEA Command: Find the Group Corresponding to the Constraint in which an Object Participates

The program prompts the user to select an object, then displays the name of the group corresponding to the constraint and highlights all objects involved in that constraint until the user presses Enter. If the object participates in multiple constraints, the program will display a list of corresponding group names for the user to select a specific group, and then highlight the relevant group. This process can be repeated as desired by the user.

8.2 The GDES Command: Display the Description String of a Constraint

First, in a manner similar to the GSEA command, the program will identify the name of the group G that the user wishes to process. Then, it will search for the xrecord with the key "Description" in the extension dictionary of G, and display the contents of the string $\mathcal{S}$ stored in that xrecord.

8.3 The HILO Command: Highlight Objects Participating in a Constraint Based on Their Order in the Constraint Description String

The program will identify the group G corresponding to the constraint in the same way as in the GSEA command, then highlight the first object. When the user presses Enter, the highlight is removed, and the process repeats for the next object, and so on.

8.4 The ECON Command: Delete a Constraint

The program finds the group corresponding to a constraint using the same approach as in the GSEA command. It then deletes objects in the group whose layers are named "ex_1", "ex_2", ... or "in_1", "in_2", ... (see Section 9.5). These are the objects that were added to the original drawing for constraint creation. Next, the program explodes the group and removes its name.

8.5 The HIOB Command

If you only want to display objects originally present in the drawing, you can use the HIOB command. This command will turn off all layers named "ex_1", "ex_2", ... or "in_1", "in_2", ... (see Section 9.5) if at least one of these layers is currently turned on and none of them is the current layer.

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 149-157, June-2025
https://doi.org/10.47001/IRJIET/2025.906019

If all these layers are already off, the command will turn them on - in other words, running the HIOB command again will display all objects that were added to the original drawing to create constraints.

IX. STRUCTURE OF THE BASE BLOCK

9.1 Name of the Base Block

The name is chosen appropriately to imply the type of constraint that the block is designed to generate.

9.2 Insertion Base Point of the Base Block

This point is selected as the starting point of the first Line object among the externally-constrained objects (see Section 9.3) of the block.

9.3 Types of Objects in the Base Block

From this section onward, the term “externally-constrained objects” of a base block refers to Line objects which, after being inserted into the drawing by inserting the block and then exploding the resulting block reference, will either be directly constrained to external objects (outside the block) or assigned additional dimensions. Such supplementary constraints are called external constraints. The remaining objects (if any) in the block are referred to as internal objects. Constraints that are pre-applied between objects within a base block are called internal constraints.

9.4 Notational Conventions

- The names of the original objects in the drawing, before constraints are applied: $\bar{a}$, $\bar{b}$, $\bar{c}$, $\bar{d}$.
- The names of externally-constrained objects: $\bar{x}$, $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$, …
- The names of internal objects: $\bar{t}_1$, $\bar{t}_2$, $\bar{t}_3$, …
- The lengths of the Line objects $\bar{a}$, $\bar{x}_i$, $\bar{t}_i$ are a, x_i, t_i, respectively.
- The starting point of the Line object $\bar{x}$ is X’.
- The endpoint of the Line object $\bar{x}$ is X”.
- The handle of the Line object $\bar{a}$ is h_a.
- The handle of the Line object $\bar{x}_i$ is h_{xi}.
- The name of the group containing the objects participating in a constraint is Gname.

These conventions will continue to be used in the following sections.

9.5 Identifying Object Types by Layer Name

Internal objects are placed on layers whose names begin with "in_", specifically $\bar{t}_1$, $\bar{t}_2$, $\bar{t}_3$, … are placed on layers "in_1", "in_2", "in_3", … in order.

Externally-constrained objects are placed on layers whose names begin with "ex_", specifically $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$, … are placed on layers "ex_1", "ex_2", "ex_3", … in order.

The default color of these layers is set to be fairly light to minimize confusion with the colors of objects in the original drawing. The HIOB command (see Section 8.5) can be used to turn all of these layers on or off.

X. SOME BASE CONSTRAINTS AND THEIR CORRESPONDING BASE FIGURES

According to the principles presented in Section II, a wide variety of base functions can be constructed by taking advantage of geometric equalities and inequalities. For illustration, several base functions and their corresponding constraints are described in detail below.

Note:

- In the figures, the red and cyan elements contain symbols for additional explanations.
- If a numeric constant needs to be input to a base function, it must be in the form of a floating-point or integer value corresponding to the Double or Long data types in VBA.
- In the base function calls listed below, if h_a is replaced by nil, there is no need to assign the constraint a = x_1 (see Section 6.1).

10.1 The Constraint a ≤ b

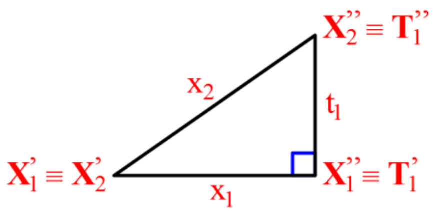

Figure 2: Base figure for the constraint a ≤ b

- **Base figure:** Figure 2
- **Constraint name:** "<="
- **Description string:** "(<= p1 p2)"
- **Objects to be entered in the INPC command:** $\bar{a}$, $\bar{b}$
- **Base function:** (Leoe Gname h_a h_b)
- **Return value:** h_{x1}
- **External constraints:** a = x_1, b = x_2
- **Internal constraints:**
  - $\bar{x}_1 \perp \bar{t}_1$
  - Several “Coincident” constraints are specified to indicate coincident endpoints of suitable pairs of line segments:

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 149-157, June-2025
https://doi.org/10.47001/IRJIET/2025.906019

$X_1' \equiv X_2',\ X_1'' \equiv T_1',\ X_2'' \equiv T_1''$

10.2 The Constraint a ≥ b

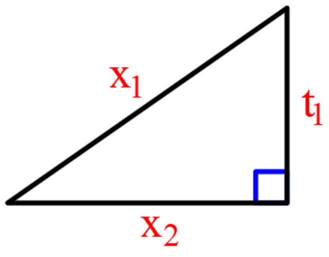

Figure 3: Base figure for the constraint a ≥ b

- **Base figure:** Figure 3
- **Constraint name:** ">="
- **Description string:** "(>= p1 p2)"
- **Objects to be entered in the INPC command:** $\bar{a}$, $\bar{b}$
- **Base function:** (Groe Gname h_a h_b)
- **Return value:** h_{x1}
- **External constraints:** $a = x_1$, $b = x_2$
- **Internal constraints:**
  - $\bar{x}_2 \perp \bar{t}_1$
  - Several "Coincident" constraints are specified to indicate coincident endpoints of suitable pairs of line segments.

10.3 The Constraint a = b + c

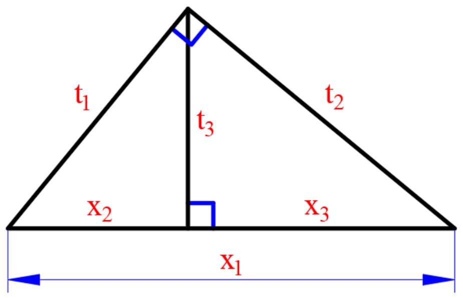

Figure 4: Base figure for the constraint a = b + c

- **Base figure:** Figure 4
- **Constraint name:** "+"
- **Description string:** "(+ p1 p2 p3)"
- **Objects to be entered in the INPC command:** $\bar{a}$, $\bar{b}$, $\bar{c}$
- **Base function:** (Addi Gname h_a h_b h_c)
- **Return value:** h_{x1}
- **External constraints:** $a = x_1$, $b = x_2$, $c = x_3$
- **Internal constraints:**
  - $\bar{t}_1 \perp \bar{t}_2,\ \bar{t}_3 \perp \bar{x}_2,\ \bar{t}_3 \perp \bar{x}_3$
  - Several "Coincident" constraints are specified to indicate coincident endpoints of suitable pairs of line segments.

10.4 The Constraint a = b – c

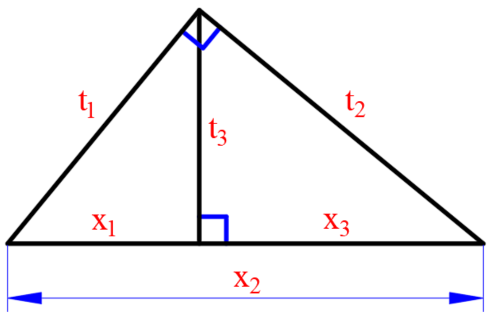

Figure 5: Base figure for the constraint a = b – c

- **Base figure:** Figure 5
- **Constraint name:** "-"
- **Description string:**"(- p1 p2 p3)"
- **Objects to be entered in the INPC command:** $\bar{a}$, $\bar{b}$, $\bar{c}$
- **Base function:** (Subtr Gname h_a h_b h_c)
- **Return value:** h_{x1}
- **External constraints:** $a = x_1$, $b = x_2$, $c = x_3$
- **Internal constraints:**
  - $\bar{t}_1 \perp \bar{t}_2,\ \bar{t}_3 \perp \bar{x}_1,\ \bar{t}_3 \perp \bar{x}_3$
  - Several "Coincident" constraints are specified to indicate coincident endpoints of suitable pairs of line segments.

10.5 The Constraint $a = \frac{b.c}{d}$

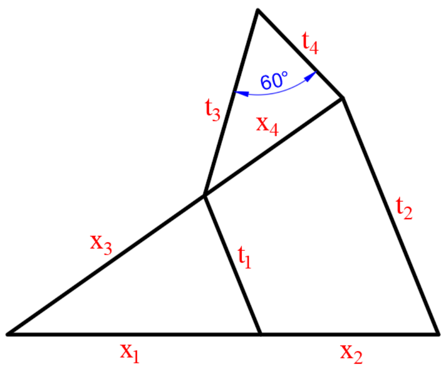

Figure 6: Base figure for the constraint $a = \frac{b.c}{d}$

- **Base figure:** Figure 6
- **Constraint name:** "*/"
- **Description string:** "(*/ p1 p2 p3 p4)"
- **Objects to be entered in the INPC command:** $\bar{a}$, $\bar{b}$, $\bar{c}$, $\bar{d}$
- **Base function:** (Mul Gname h_a h_b h_c h_d)

- **Return value:** h_{x1}
- **External constraints:** $a = x_1,\ b = x_2,\ c = x_3,\ d = x_4$
- **Internal constraints:**
  - $\bar{t}_1\ //\ \bar{t}_2$
  - The angle between $\bar{t}_3$ and $\bar{t}_4$ is 60° (to ensure that $x_4 \neq 0$).
  - Several "Coincident" constraints are specified to indicate coincident endpoints of suitable pairs of line segments.
  - Several "Colinear" constraints are applied to suitable pairs of Line objects, namely the pairs $\bar{x}_1$ and $\bar{x}_2$, and $\bar{x}_3$ and $\bar{x}_4$.

10.6 The Constrainta $= \sqrt{b\,.\,c}$

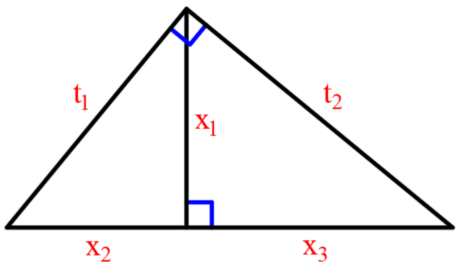

Figure 7: Base figure for the constraint a $= \sqrt{b.c}$

- **Base figure:** Figure 7
- **Constraint name:** "gm"
- **Description string:** "(gm p1 p2)"
- **Objects to be entered in the INPC command:** $\bar{a},\ \bar{b},\ \bar{c}$
- **Base function:** (Gmean Gname h_a h_b h_c)
- **Return value:** h_{x1}
- **External constraints:** $a = x_1$, $b = x_2$, $c = x_3$
- **Internal constraints:**
  - $\bar{t}_1 \perp \bar{t}_2,\ \bar{x}_1 \perp \bar{x}_2,\ \bar{x}_1 \perp \bar{x}_3$
  - Several "Coincident" constraints are specified to indicate coincident endpoints of suitable pairs of line segments.

10.7 Special Cases

The following section examines some special cases that occur when, in the constraints presented in Sections 9.1 to 9.6, certain variables are replaced with numeric constants (denoted as λ, μ). Specifically, we consider the cases:

$$a \le \lambda,\ \ a \ge \lambda,\ \ a = b + \lambda,\ \ a = b - \lambda,\ \ a = \frac{\lambda.b}{\mu},\ \ a = \frac{\lambda^2}{d},$$

$$a = \sqrt{\lambda\,.\,b}$$

These special cases can use the same base blocks as the general case. When the INPC command is executed, the user enters a constraint description string in which some parameter names are replaced with string representations of numeric constants. The base function will then assign an "Aligned" dimension constraint (in AutoCAD) to set the length of certain Line objects equal to the entered numeric constant, treating it as an external constraint.

It is also possible to create a simpler base figure (if available), assign a new name to the constraint, and subsequently define a new base function for the special case.

When it is necessary to assign a dimensional constraint that sets the value of the length variable x equal to a numeric constant λ, but λ is either very small or very large compared to the lengths of the line segments in the current drawing, then, to better facilitate user observation or accurate computation by AutoCAD's constraint solver, the following two methods can be used in many cases:

- Transform the constraint into an equivalent form that avoids excessively small or large numeric constants, as described above.
- Before assigning any constraints, including those already available in AutoCAD, apply the SCALE command to the entire drawing using a scaling factor $\eta > 0$ to produce a new version of the drawing. This is equivalent to choosing a different unit of measurement for lengths. Geometric design will then be carried out on the scaled drawing. In many cases, η can be chosen such that the constraints no longer involve excessively small or large numeric constants.

10.8 The Constraint a = λ (Where λ Is a Non-Negative Numeric Constant)

Figure 8: Base figure for the constraint a = λ

If the object $\bar{a}$ already exists in the initial drawing, it is sufficient to apply a dimensional constraint type already available in AutoCAD.

The base function in this section applies to the case where the object $\bar{a}$ does not currently exist in the original drawing but will be created by another constraint that returns the handle h_a.

In the constraint description string, at the position corresponding to $\bar{a}$, the description string of that other constraint will be used instead.

- **Base figure**: Figure 8
- **Constraint name**: "=c"
- **Example of a constraint description string using this constraint**:
  "(=c 10 (+ nil (gm p1 p2) (gm p3 p4)))"

This string corresponds to the constraint:

$\sqrt{p_1p_2} + \sqrt{p_3p_4} = 10$

- **Objects to be input in the INPC command**: $\bar{a}$
- **Base function**: (Eqc Gname λ h_a)
- **Returned result**: h_{x1}
- **External constraints**: $a = x_1$, $x_1 = \lambda$
- **Internal constraints**: None

XI. DISCUSSION AND CONCLUSION

The capability of AutoCAD to create and manage 2D parametric constraints has enhanced the modularity, flexibility, and real-time interactivity of models, while reducing the need for manual modifications during the design process. The method proposed in this paper further extends the ability to generate constraints for models and, moreover, enables the creation of bidirectional constraints. A wide range of base constraints can be generated by leveraging the rich set of equalities and inequalities in planar geometry. This constraint construction approach is also useful when a complex constraint can be replaced by a system composed of constraints available in AutoCAD and the base constraints established and utilized according to the method described above.

REFERENCES

[1] B. Bettig and C. M. Hoffmann, "Geometric constraint solving in parametric CAD," *Journal of Computing and Information Science in Engineering*, vol. 11, no. 2, p. 021001, 2011, doi: 10.1115/1.3593408.

[2] P. Company, F. Naya, M. Contero and J. D. Camba, "On the role of geometric constraints to support design intent communication and model reusability, "*Computer-Aided Design and Applications*, vol. 17, no. 1, pp. 61-76, 2020, doi: 10.14733/cadaps.2020.61-76.

[3] R. Ellis, "A Practical Guide to Parametric Drawing in AutoCAD," Autodesk University handout, class AS2503-L, Autodesk, Las Vegas, NV, USA, 2024.

[4] Autodesk, "AutoCAD 2025 Help," online documentation, 2024. [Online]. Available: https://help.autodesk.com/view/ACD/2025/ENU/. [Accessed: 09-May-2025].

[5] M. Cottingham, *Mastering AutoCAD VBA*, Sybex Inc., Alameda, CA, USA, 2001.

Citation of this Article:

Pham Tuan Anh. (2025). Enhancing AutoCAD with the Ability to Create Bidirectional Equality and Inequality Constraints on Line Segment Lengths. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(6), 149-157. Article DOI https://doi.org/10.47001/IRJIET/2025.906019

****