The `muac` software family handbook

Mariano Vázquez Espí

Madrid, April 10, 2017.

Abstract

`muac` is the acronym of *Método Universal de Análisis de Cerchas*, i.e., Truss-like structures Universal Analysis Method—`tuam`?—. The program manages different kinds of computation problems for structures with only tension/compression members. This handbook is incomplete and for the time being it covers only most frequently used functions. `muac` has some useful companions: `muacad`, `muacmlatex` and others. They take the `muac` results and translate it to the languages of AutoCAD[1] or LaTeX, for example, ready to be include in drawings or text documents. Hence, `muac` has no graphic output by itself. Furthermore, `muac` does not ask any question to the user (as it seems appropriate in the communication between an inferior being with a human being).

Contents

List of Figures

List of Tables

[1] AutoCAD is a trade mark from Autodesk Inc.

1. Distribution

`muac` is distributed for the time being only as binary files for Generic GNU/Linux kernels along with this handbook and related code files (`muac.tgz` file). However you can send your problem to `sce` at `ee.upm.es` and, providing that your message arrives from a real mailbox, you will receive some answer including results for the problem you pointed out or description of any error condition, or both. For no Linux users, this handbook and other useful files are packed in a short distribution (`muac.zip` file)

2. A short description

`muac` reads from the standard input the description of some structure under loading, make the indicated computation, and print the input and the results to the standard output. See the documented options in TABLE 4

The terminology adopted follows from the description of Universal Method in the case of proportional (or linear) state of the structure such this is taught in School of Architecture of Madrid (UPM), and you can see it in *Sólido deformable (II)* (in Spanish). The language that `muac` understands is very simple: a list of lists, each of which is an alphabetic character followed by a list of numbers. Numbers and characters must be separated with white space (including carriage-return, tabs, etc). The meaning of each character is shown in TABLE 1. All input after one `#` up to the end of the line is ignored (this mechanism is included to introduce easily comments in input and output data).

TABLE 1: ALPHA-CHARACTERS THAT INTRODUCE LISTS FOR `muac` AND FAMILY

Conventions for both, input and output files		
Char:	element of the list	
`B`	: bars as *initial-node final-node cross-section-type*	three integers per bar.
`C`	: nodal coordinates: *x-coordinate y-coordinate*	two rationals per node.
`D`, `d`	: nodal displacements: *x-displacement y-displacement*	
`E`, `e`	: bar unitary strains: *strain*	
`H`, `h`	: exertions: *bar-exertion*	i.e., internal force of the bar, axial force.
`M`	: materials: *Young's-modulus*	one rational per material.
`P`	: action loads: *node unsigned-value angle*	*unsigned-value* is the load modulus; *angle* with x-axis;
	:	one integer and two rationals per load.
`Q`	: action loads: *node axis signed-value*	*axis* is 1 (x) or 2 (y);
	:	two integers and one rationals per load.
`R`, `r`	: reactions: *node axis signed-value*	
`S`	: cross-section types: *material-type area*	integer and rational per type.
`V`	: displacement conditions: *node axis displacement*	displacement null for theoretic support;
	:	two integers and one rational per condition.
`X`	: matrix $\mathbf{B}$	by rows.

The list of nodal coordinates must be the first list in the input. Each node can be referenced after by its ordinal position in the nodal list. This implicit numbering schema is used too in material list and section type list.

The only definitely defined sign convention is this: traction and elongation are considered as positive, and compression and shortening, as negative. The axes xy can be freely chosen, the units are free also. It is suggested to use the implicit axes as the standard ones in mathematics: positive y to up, positive x to right, and angles counter clockwise from positive x-axis; this convention is used hereafter.

2.1. A simple example

FIGURE 1 shows a simple example of a truss-like structure (or framework, or bent). There are two support points: pin and rolled respectively. There are three loads and five bars of the same material but different cross-section areas. TABLE 2 shows the input data for `muac` and TABLE 3 the corresponding output. The command for producing the latter is simply:

```
muac < ejemplo1.muac > ejemplo1.out.muac
```

Providing that the data in TABLE 2 is stored in file `ejemplo1.muac`, then the results of TABLE 3 is a part of the contents of the output file `ejemplo1.out.muac`.

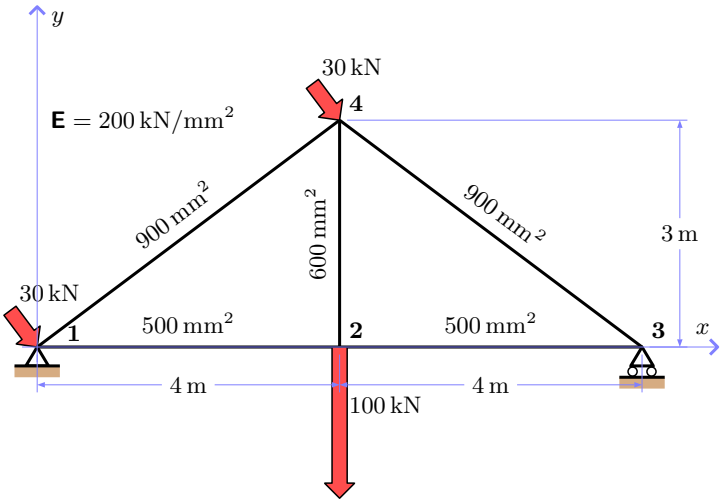


FIGURE 1: A SIMPLE EXAMPLE

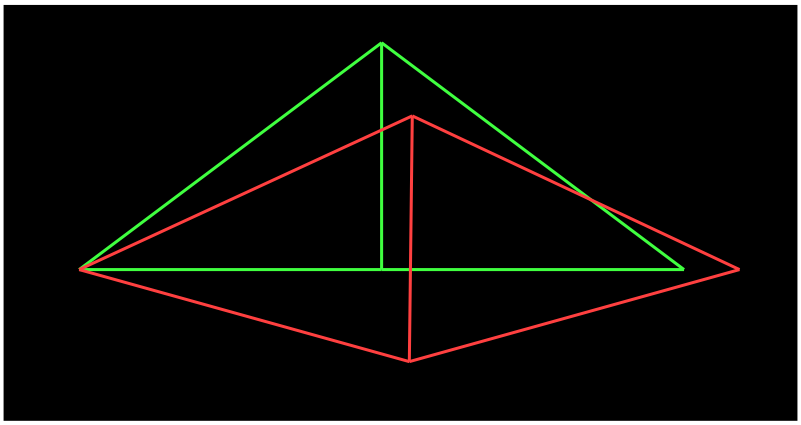


FIGURE 2: DRAWING BY `muacad`

2.2. `muacad`

`muacad` reads from the standard input and if the reading corresponds to the description of some structure under loading with the results of `muac`, it prints to the standard output instructions in the script language of AutoCAD, that can be stored in a file and then read by AutoCAD. Two special layer must be previously defined in the AutoCAD drawing, with the names defined by `muacad` or by the user. `muacad` has several options, see TABLE 5. The results for the example of FIGURE 1 is shown in FIGURE 2. The script for AutoCAD can be produced with commands like:

```
muac < ejemplo1.muac > ejemplo1.out
muacad < ejemplo1.out > ejemplo1.scr
```

or with a pipe between the two programs:

```
muac < ejemplo1.muac | muacad > ejemplo1.scr
```

Once the `scr` file is ready, it must be read from AutoCAD drawing.

2.3. Preparing the input with AutoCAD

With the LISP functions into `muac.lsp`, part of the input file can be get from the truss drawing in AutoCAD. Further more, the output file can be directly read, without using `muacad`.

From the command line of AutoCAD:

TABLE 2: DATA EXAMPLE FOR muac
Data for the simple example of FIGURE 1

```
# simple example
C  # in meters
    0 0   4 0   8 0   4 3
B  # node node section-type
    1 2 1   2 3 1 # ties
    2 4 2         # king post
    1 4 3   3 4 3 # rafters
M  # normal steel
    200 # E=kN/mm2
S  # material type and areas in square milimeters
    1 500 # ties
    1 600 # king post
    1 900 # rafters
P  # loads in kN and degree
    1 30 -53.13
    4 30 -53.13
Q  # loads in kN
    2 2 -100
V  # constrained displacements
    1 1 0   1 2 0   3 2 0
```

```
(load "muac.lsp")
```

Once loaded, the following commands are defined:

MUAC command writes a file that can be read by `muac` from the previously selected lines. The cross-section code of each bar is set as its line colour code (bylayer=1, red=1, yellow=2, etc). The remaining data (sections, materials, loads, displacement conditions) must be added editing the file thus generated.

MUAC-READ command reads a `muac` output file, and generates internal data structures with these results.

Once you have read a `muac` output file with the `MUAC-READ` command, the following commands are available:

NUMERATE draws the node numbers

EXERTIONS draws the internal forces of the bars by using a colour code: from lowest (red) to highest (blue) values. Tensions are drawn with a continuous line, compressions are drawn with dashed lines.

QUERY-BAR: After pointing a bar in the screen, `QUERY-BAR` indicates the value of the internal force of the designated bar.

3. The `mail` interface

For using `muac` and related software by `mail`, you must send a message from a real mailbox to `sce` at `ee.upm.es` with the following guidelines:

`Subject:` *label*
program $program_{options}$ `<<\EOF`
program-data
`EOF`

with the following meanings:

label an alphabetic label that will be used to label the produced files that will be send to you as attached files into the reply message. Only classical Latin letters (and not `ñ`) are allowed in this label: if you need numbers use roman numbers (`i`, `ii`, etc).

program one of `muac` or `muacad`

$program_{options}$ optionally, you can include some options for the *program* (see TABLE 4 or TABLE 5).

TABLE 3: RESULTS FROM muac
Results from muac for the simple example of FIGURE 1 with the input data of TABLE 2

```
# simple example
C  # in meters
    0 0   4 0   8 0   4 3
```

(some input data suppressed here)

```
# muac RESULTS

# 5 equations
# bar 1: Lx= 4; Ly= 0; EA= 100000.
# bar 2: Lx= 4; Ly= 0; EA= 100000.
# bar 3: Lx= 0; Ly= 3; EA= 120000.
# bar 4: Lx= 4; Ly= 3; EA= 180000.
# bar 5: Lx= -4; Ly= 3; EA= 180000.
# equation 1 pivot: 50000
# equation 2 pivot: 40000
# equation 3 pivot: 35540
# equation 4 pivot: 31143.5
# equation 5 pivot: 13488.8
# Bar strains
E
0.000916667 # bar 1 cross-section 500
0.000916667 # bar 2 cross-section 500
0.000833333 # bar 3 cross-section 600
-0.000511574 # bar 4 cross-section 900
-0.000636574 # bar 5 cross-section 900
# Bar exertions
N
91.6667 # bar 1 with cross-section 500 (def:0.000916667, E:200)
91.6667 # bar 2 with cross-section 500 (def:0.000916667, E:200)
100 # bar 3 with cross-section 600 (def:0.000833333, E:200)
-92.0833 # bar 4 with cross-section 900 (def:-0.000511574, E:200)
-114.583 # bar 5 with cross-section 900 (def:-0.000636574, E:200)
# Reactions: node axis signed-value
R 1 1 -3.60e+01
1 2  7.92e+01
3 2  6.88e+01
# displacements: axes 1 and 2
D 0 0 # node 1
0.00366667 -0.0121728 # node 2
0.00733333 0 # node 3
0.00405729 -0.00967284 # node 4
```

program-data the description of the structural problem; in the case of `muacad` normally it is the output of `muac`

Normally, you will receive a message with the results as attached files, with the following name schema: *label-your-email-year-month-day-hour-minute-program*`.`*extension*, where *extension* would be one of `out` (the output of *program*) and `err` (the standard error output of the *program*, if any).

The messages received are queued and processed according to the general rules of the GNU/Linux `batch`.

It has no sense to reply to the messages received from `sce` mailbox as there is no human person behind scene.

Table 4: `muac` options

option	function
`-b`	print matrix $\mathbf{B}'$: $\varepsilon = \mathbf{B}'\mathbf{g}$
`-d`	debug
`-h`	print a help message and do nothing
`-t`	check the input and copy it in the output; do nothing

Table 5: `muacad` options

option	function
`-e` *layer-name*	name of the equilibrium layer (default: `muac_ge`)
`-f` *number*	stretching factor for displacements
`-h`	print a help message and do nothing
`-i` *layer-name*	name of the initial geometry layer (default: `muac_gi`)

4. Collaborative work

Any help to continue the development of this software family will be kindly welcome and acknowledged. You can consider work around:

- formulating of new algorithms
- writing code
- writing new interfaces for other programs as `geogebra`, `maxima`,...; or other formats as AutoCAD DXF, CSV,...
- checking systematically the programs, searching for errors running them or in their documentation
- sending comments or point out malfunction or mistakes, suggesting improvements of the `muac` language, etc
- improving my no very good English

5. Important note

You can use any options of the program `muac` that has not documented here at your own risk. Actually, `muac` can do many others things but they are not checked seriously for now (these things are for research purposes only). In the future, I hope this handbook become more complete...

6. Acknowledgements

⋆ Antonio Marchesi wrote in the nineties a BASIC version of the prehistoric prototype of `muac` for a popular CASIO calculator at that time.
⋆ Carlos Olmedo wrote in 2016 the AutoCAD scripts, included into the 2017 distribution.
⋆ Joaquin Antuña checked the AutoCAD scripts in 2016 and 2017.
⋆ Zhou Cui corrected the English spelling of this manual.

7. History

The first C version was wrote in Madrid (1991) and it remained unchanged for more than ten years. Most improvement not documented here was wrote since 2009 to 2012 at Villamanta, Madrid, Ondara and Barcelona.

The first version of this handbook was wrote in 2015, between Madrid and Ondara (in the past, the only documentation was the heading files of the C code).

The 2017 version includes now `muac.lsp` and the second edition of this handbook. The `P` list for loads is new.

A. Errors and Warnings

`muac` and companions

`muac: [char] list missing` The "char" list is needed and is missing

`muac: Error into [...] list` If you wrote the referenced table, you perhaps wrong the type of data or included more or less elements than adequate. If `muac` or companions wrote the table: oops? perhaps a corrupted file?

`muac: Unknown alphanumeric character for introduce a list: [...]` The letter indicated is not known in `muac` semantics.

`muac: List X has too many numbers!` Very rare as the X-list is wrote by `muac`: maybe a corrupted file?

`muac: semantic error` Something in the input is not a list neither a comment...

`muac: Table [...] before table C` Table `C` must be the first table in the input

`muac: warning: modulus must be a positive number: fixed` A modulus in a `P` list was signed: it gets absolute value

`muac: warning: P and Q lists missing: we proceed anyway` Load definition missing (perhaps knowingly?)

Other messages from `muac` and companions are self-explanatory.

`sce` service

`bad options in ...` The syntax of the options for the selected program is wrong

`I can't find the EOF: request rejected!` After the first `EOF`, the second one is not found

`malformed label in Subject:...` Remind than only Latin letters are allowed in *label*: no numbers, white space, etc. Castilian speakers: *¡Ninguna ñ!*

`malformed message: request rejected!` The mail received is not conform with RFC standard: see the first line of `message.in`

`unknown program: [...] sce` has nothing to do with it

B. Revision history

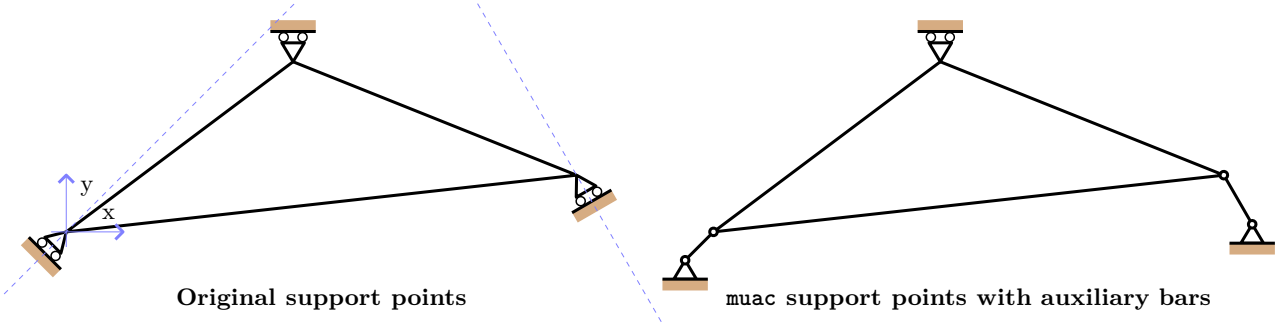
2015–03–22: Problem managing the `X` list in input: fixed.
2017–03–31: Problem managing load and reaction simultaneosly acting on a node: fixed.
2017–04–02: Problem managing reaction list by `muac`: fixed.

C. Drawbacks and how to get around them

- **Only a set of loads can be managed in input.** Working on Linux, `make` can be used to make the `.muac` file for each set. To this end, each section on a `muac` input file can be saved in a separate file: `.geo` for geometry (nodes and bars), `.load` for load data, `.thick` for material and cross-section data, etc. Then, `make` can be commanded (via a `makefile`) to make each `.muac` file for each different `.load` file, and processing it with `muac` and others as appropriate. In other operating systems with `make` (or similar program), the last step can be substituted for sending a mail to the `sce` service.

  The previous approach is not by far the only possible. As the load lists are read accumulatively, the alternative is to make a `.muac` file for each needed combination of loads from the `.load` files. Again, `make` (or a script shell as well) can do the work.

- **Only two axes can be used for all displacement conditions.** If only a point is a rolled support on a plane nor vertical neither horizontal, one always can adopt these axes; but this approach will fail for two rolled support with different axes. A better, general, pedagogic approach is modelling these special cases with auxiliary bars with appropriate stiffness and geometry connecting with auxiliary pin supports, in such a manner that the displacement of the point only can be in the intended direction.



Original support points **`muac` support points with auxiliary bars**

With this approach, the stiffness of the auxiliary bars can simply be very very large, or can be of the value appropriate for modelling the stiffness of the foundation on the ground.

D. The gory details

Integer data: It results to error if a rational number is read when an integer is expected (v.g.: index of other list).

Rational data (floats): The rational numbers can be wrote in standard C convention: see any `muac` results as an example: `muac` and companion can read any output from `muac`.

Lists which can be used only once: `C`, `B`, `M`, `V` and `S`. If any of them is used many times, the results are undefined.

Load and reactions lists: It can use these list as many times as needed: the loads of a list will be added to those of previous one. However, `muac` ignores the reactions list in input, if any.

Load angle: It must be in degrees, not radians.

Comments: Comments from `muac` have useful information about the truss. Unfortunately, while I will try to maintain its actual format, the back-compatibility of future versions cannot be guaranteed.