UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Ingegneria
"Enzo Ferrari"

MILLE CHILI

FEM Fundamentals and Chassis Design

Sara Mantovani
sara.mantovani@unimore.it

Agenda

- Introduction to Maxima
- Maxima operators
- References
- Castigliano’s Theorem
- References

Agenda

- **Introduction to Maxima**
  - Download Maxima
  - Open, close and save Maxima file in Linux
  - The main toolbar
- Maxima operators
- References
- Castigliano’s Theorem
- References

Introduction

Maxima is a system for the **manipulation of symbolic** and **numerical expressions**, including:

- differentiation,
- integration,
- Taylor series,
- Laplace transforms,
- ordinary differential equations,
- systems of linear equations,
- polynomials,
- vectors, matrices and tensors.

Maxima yields high precision numerical results by using exact fractions, arbitrary-precision integers and variable-precision floating-point numbers.

Maxima can plot functions and data in two and three dimensions.

Maxima
Download

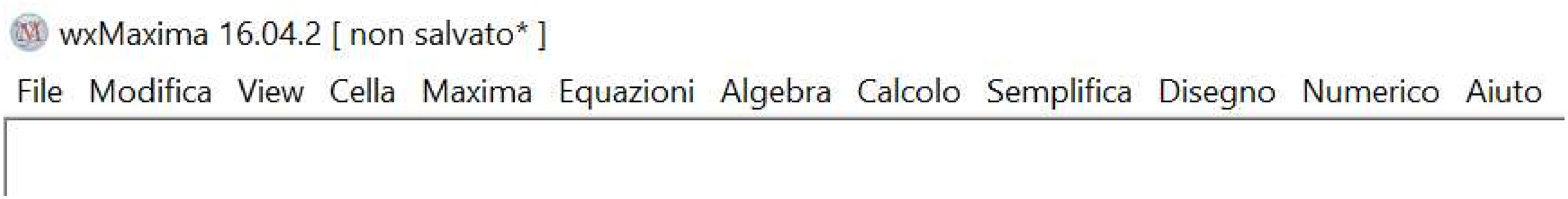
1. In www.google.it
2. Find the string data: maxima cas
3. Select the first website or alternatively move directly to the link http://maxima.sourceforge.net
4. Download the Maxima version (Windows, Linux, IOS) coeherent with your PC operating system.
5. Finally, install the program following the instructions.

NOTE: For IOS, the version Maxima 5.36.1 is surely working; althougth, this version is not the most recent version.

Open Maxima in Linux

To invoke Maxima:
1) from UNIMORE LAB PC
- Browse and run installer program
- Education
- Maxima Algebra System

2) in a console:
- type maxima and then <enter>



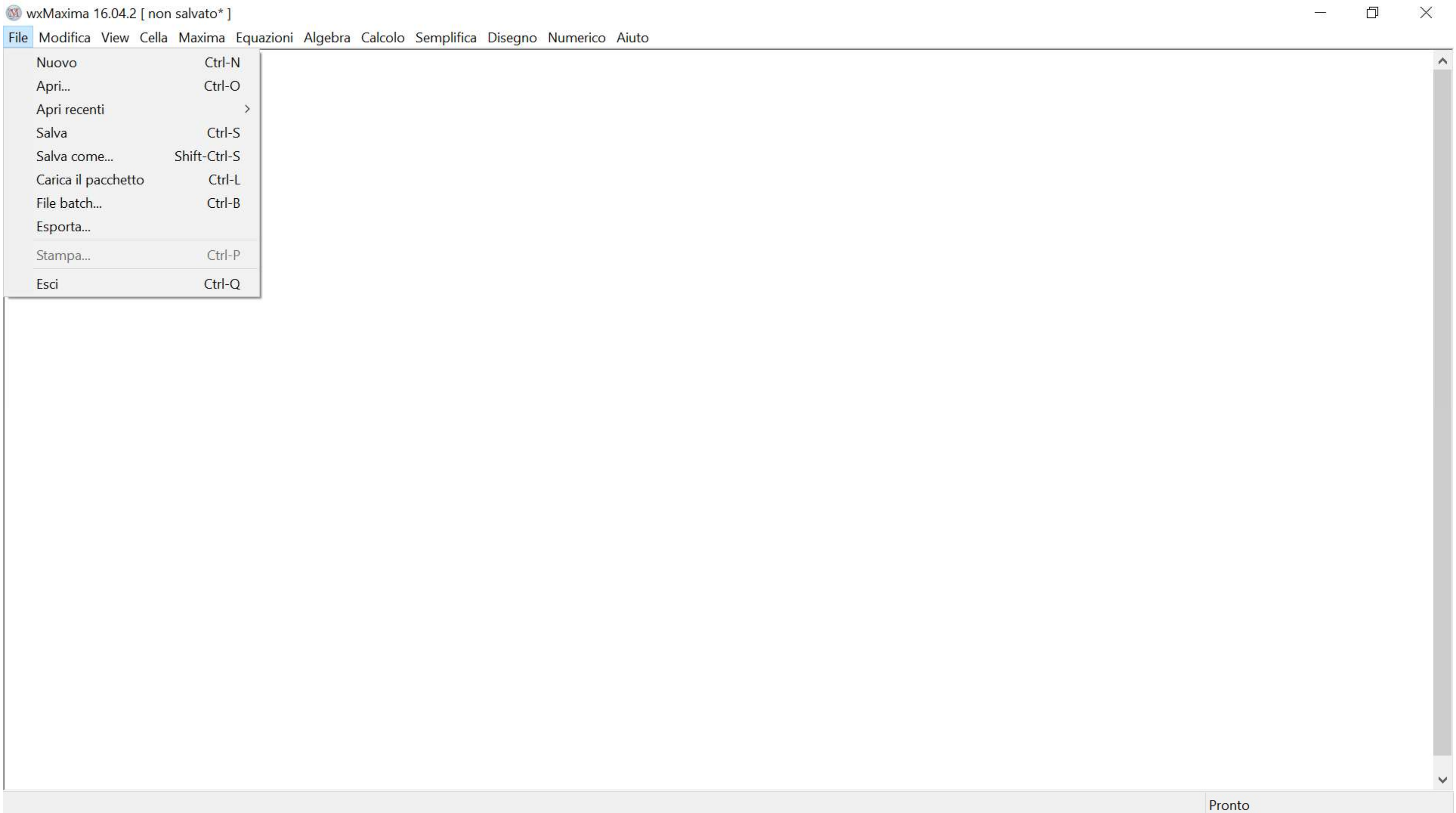
Close and save Maxima in Linux

To exit Maxima:
1) from UNIMORE LAB
- type quit()

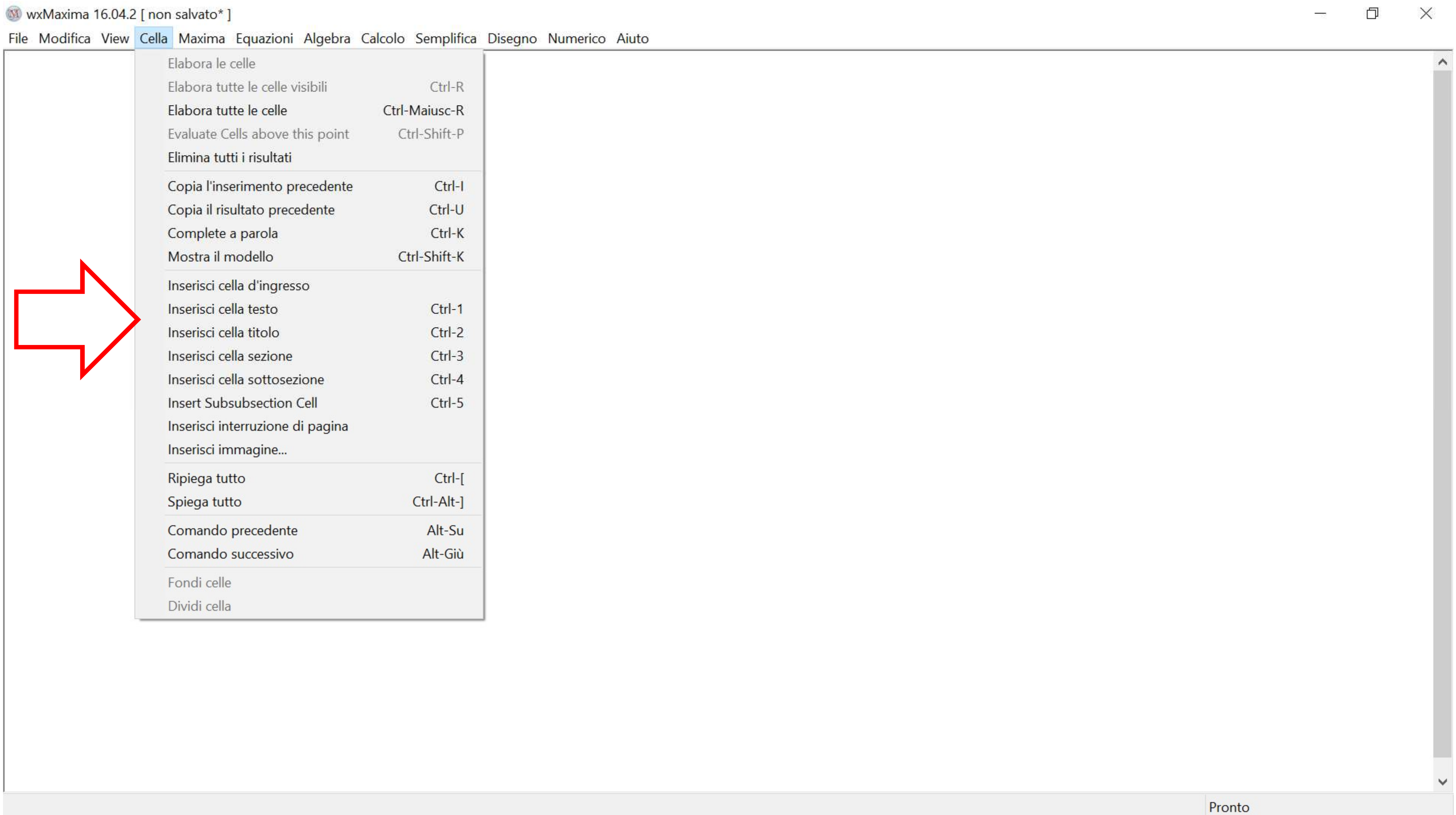
2) in a console:
- File
- Exit or CTRL+Q

Maxima files are saved as `.wxmx`

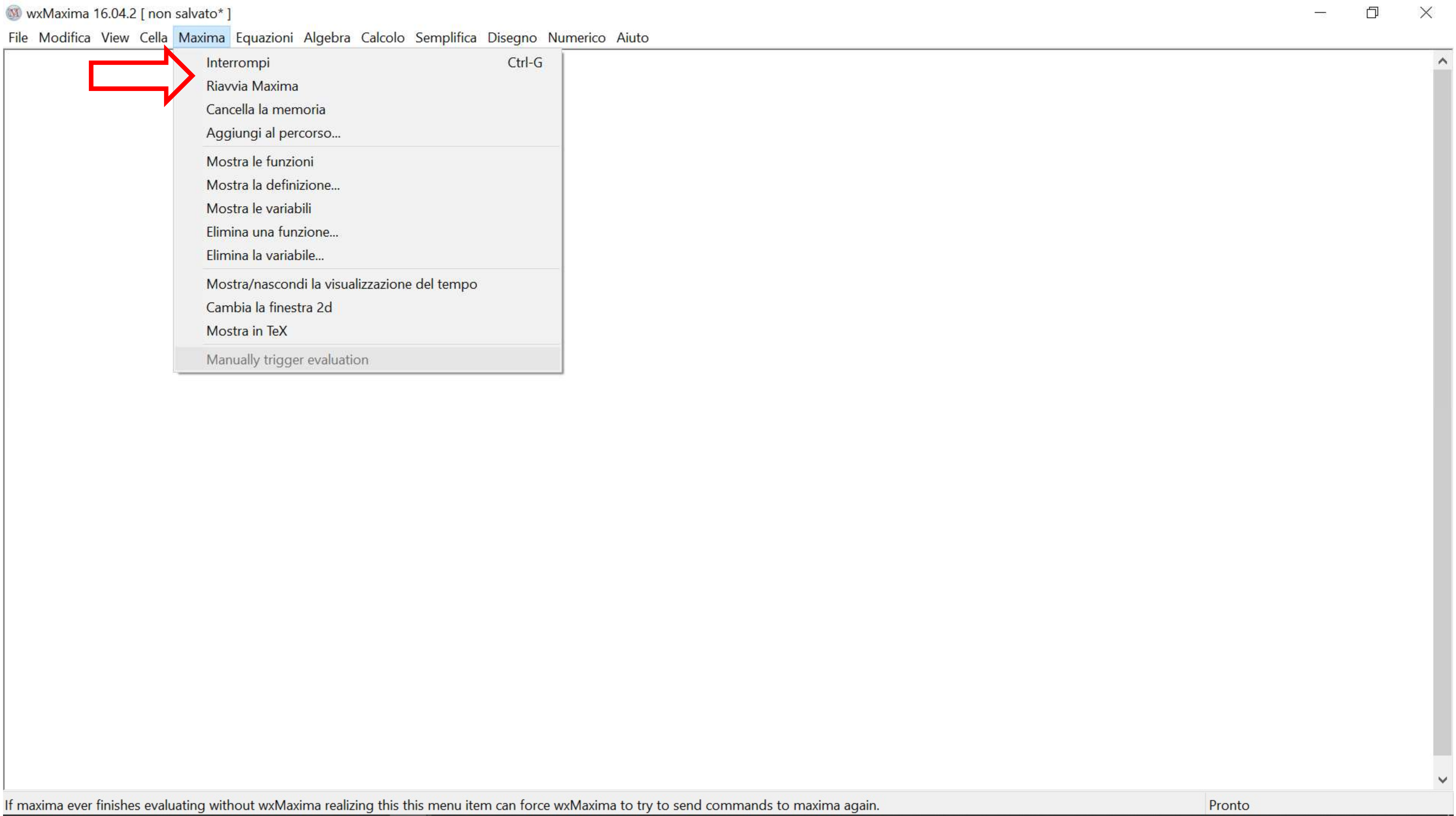
The Main Toolbar



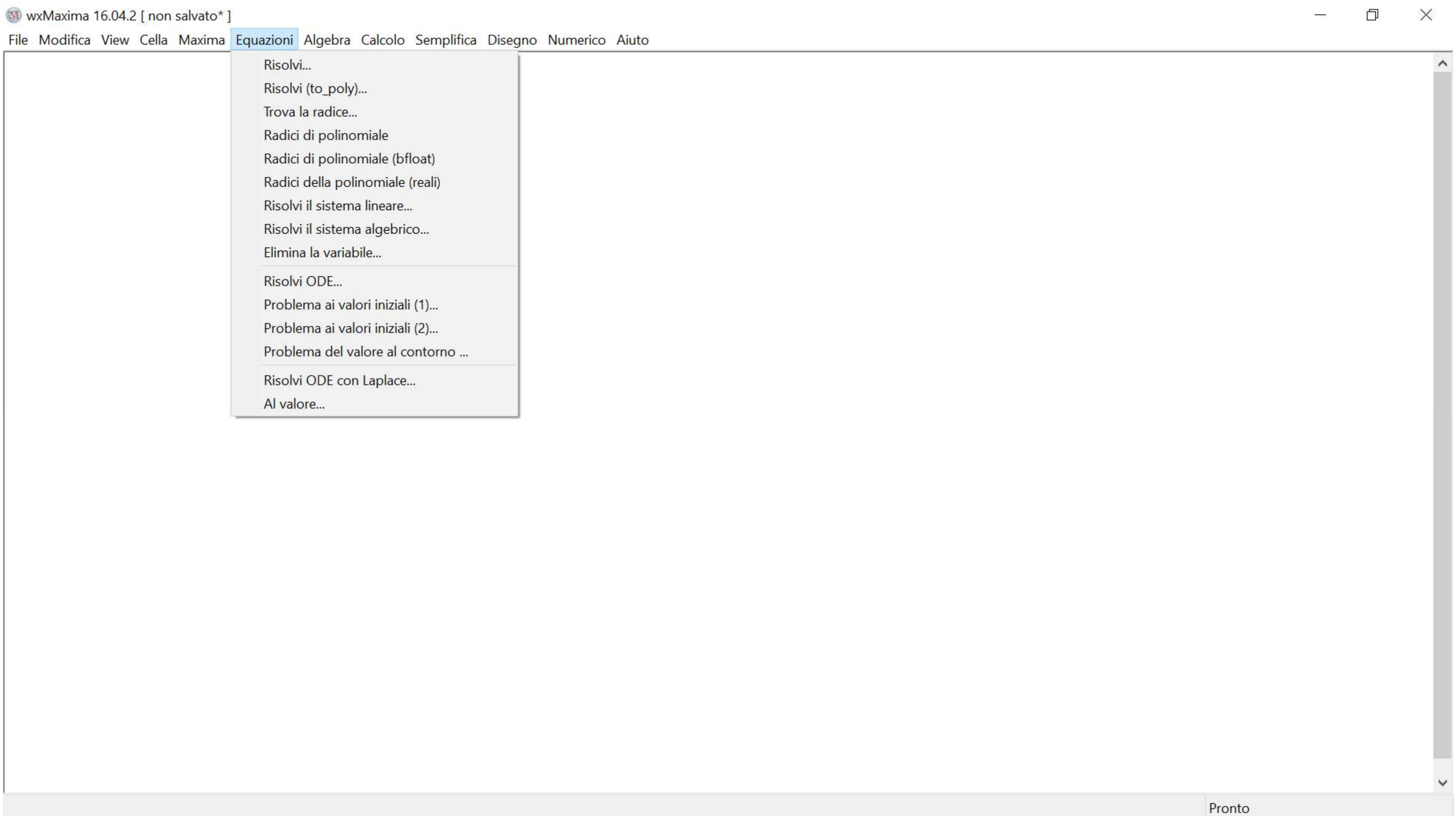
The Main Toolbar



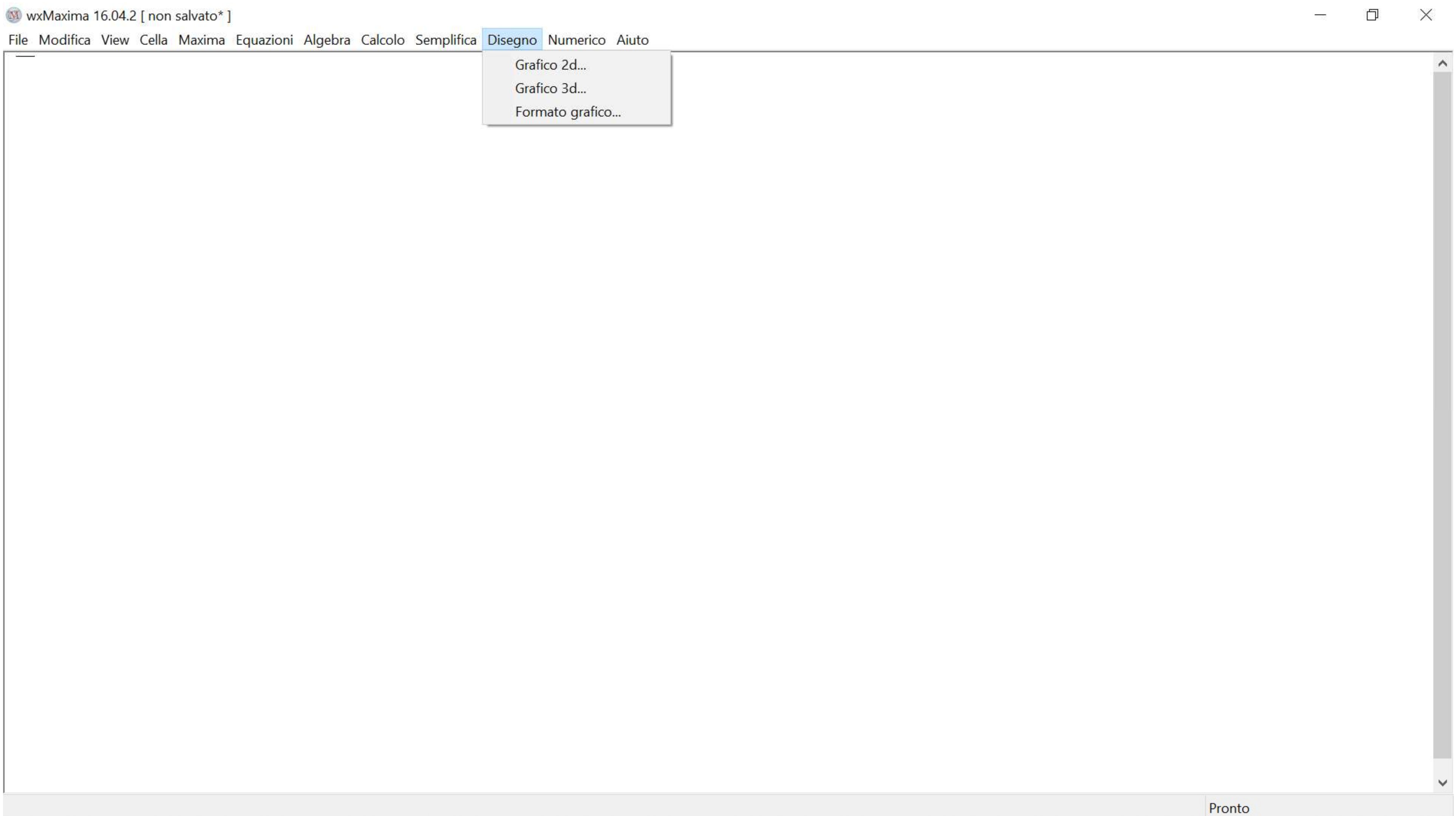
The Main Toolbar



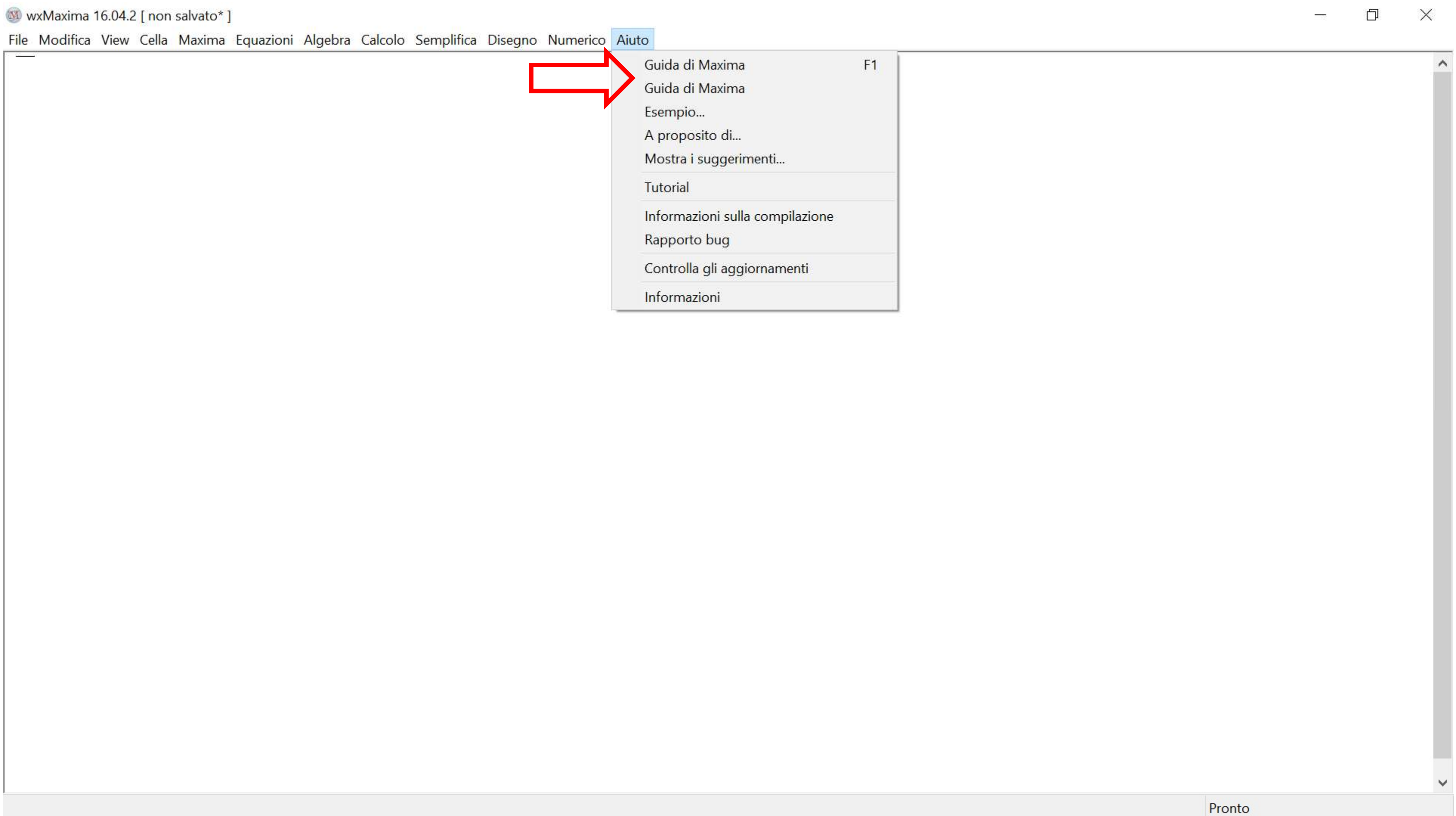
The Main Toolbar



The Main Toolbar



The Main Toolbar



Agenda

- Introduction to Maxima
- **Maxima operators**
  - Input and output
  - Starting function `kill()`
  - Terminator and special characters
  - Assignment (`:`)
  - Equation (=)
  - Function (`:=`)
  - Declarations `assume()` and `define()`
  - Numerical evaluation `numer` and `ev()`
  - Arithmetic operation and trigonometric functions
  - Differentiation and integrals `diff()` and `integrate()`
  - Polynomials
  - System of equations `solve()` and `linsolve()`
  - Two-dimensional plot `plot2d()`
- References
- Castigliano's Theorem
- References

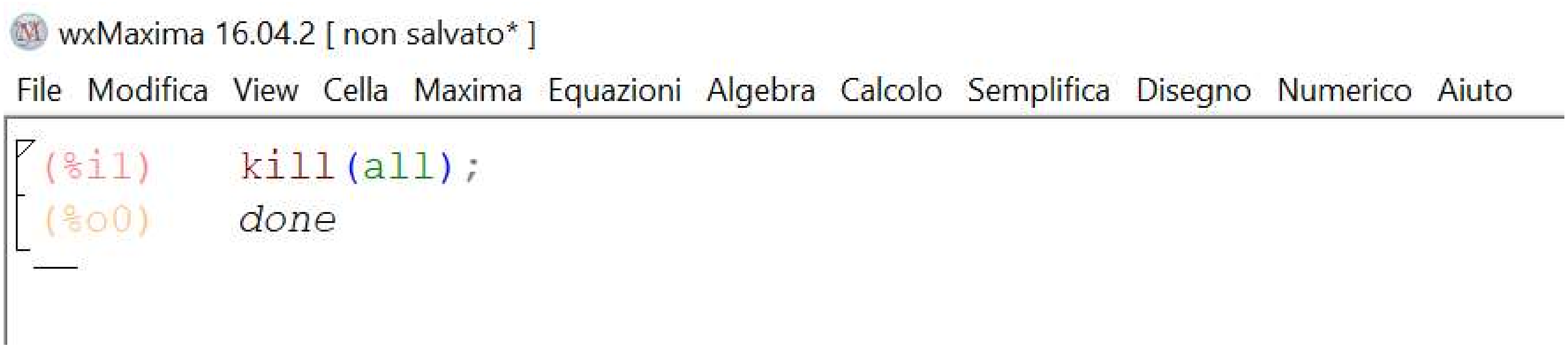
Maxima operators

Input(%i#) and output (%o#)

Beside the prompt **(%i#)** the operation might be defined.

Any input must be closed by the semicolumn character (;)

The prompt **(%o#)** represents the operation output.



NOTE: Maxima is a case-sensitive program therefore as general rule, we suggest to adopt any command/operation/variables in lowercase letter alone.

Starting function

`kill()`

Operators	Symbol
Unbinds all the items in all the infolists.	`kill(all);`
Removes the variable **a** with all its assignments and properties	`kill(a);`

To compute:
- a single operation use CTRL+enter;
- all the instructions from the beginning to the end of the program adopt CTRL+R

Terminator and special characters

Operators	Symbol
Input terminator	`;`
Input terminator, which suppresses the display of Maxima's computation. *This is useful if you are computing some long intermediate result, and you don't want to waste time having it displayed on the screen.*	`$`
If you want to refer to the immediately preceding result computed by Maxima	`%`
e is the natural log base	`%e`
i is the square root of -1	`%i`
π is equal to 3.14159…	`%pi`

Examples

```
(%i1)    kill(all);
(%o0)    done

 Input terminator ; or $

(%i1)    a;
(%o1)    a

(%i2)    a$

(%i3)    b;
(%o3)    b
```

```
 Special characters %, %pi, %e, %i
 and numer

(%i4)    %;
(%o4)    b

(%i5)    %pi;
(%o5)    π

(%i6)    %pi, numer;
(%o6)    3.141592653589793

(%i7)    %e;
(%o7)    %e

(%i8)    %e,numer;
(%o8)    2.718281828459045

(%i9)    %i;
(%o9)    %i
```

Operators	Symbol
Allow the numerical evaluation of an expression in floating point	`%, numer;`

Agenda

- Introduction to Maxima
- **Maxima operators**
  - Input and output
  - Starting function `kill()`
  - Terminator and special characters
  - Assignment (`:`)
  - Equation (=)
  - Function (`:=`)
  - Declarations `assume()` and `define()`
  - Numerical evaluation `numer` and `ev()`
  - Arithmetic operation and trigonometric functions
  - Differentiation and integrals `diff()` and `integrate()`
  - Polynomials
  - System of equations `solve()` and `linsolve()`
  - Two-dimensional plot `plot2d()`
- References
- Castigliano's Theorem
- References

Assignment

: = :=

Operators	Symbol
To assign a value to a variable use *column sign*, NOT the equal sign	:
The *equal sign* is used for representing equations NOT an assignment!	=
A function definition *e.g.* f(x)	:=

Assignment operator (:)

To a simple variable

When the left-hand side is a simple variable **:** evaluates its right-hand side and associates that value with the left-hand side.

```
(%i10)   a;
(%o10)   a

(%i11)   a:10;
(a)      10

(%i12)   a;
(%o12)   10
```

The value of `10` is associate and therefore assigned to the variable `a`.
After the assignment, the variable `a` is associated to the value `10`.

Assignment operator (:)

To an element of a list

When the left-hand side is a subscripted element of a list, a matrix, an array, the right-hand side is assigned to that element. The subscript must name as existing element.

```
(%i12)  b: [3,pippo,3/5];
(b)     [3,pippo,3/5]

(%i13)  b[3];
(%o13)  3/5

(%i14)  b[3]: puffo;
(%o14)  puffo

(%i15)  b;
(%o15)  [3,pippo,puffo]
```

Lists are the basic building block for Maxima. Lists are sequence containers that allow constant time insert and erase operations anywhere within the sequence, and iteration in both directions.

Assignment operator (:)

Multiple assignment

When the left-hand side is a list of simple and/or subscripted variables, the right-hand side must evaluate to a list, and the elements of the right-hand side are assigned to the elements of the left-hand-side, in parallel.

```
(%i19)  [c, d, e]:[20,pluto,-3/77];
(%o19)  [20,pluto,-3/77]
(%i20)  c;
(%o20)  20
(%i21)  d;
(%o21)  pluto
(%i22)  e;
(%o22)  -3/77
```

Equation operation (=)

Comparison with the assignment operator (:)

```
(%i10)  a;
(%o10)  a

(%i11)  a:10;
(a)     10

(%i12)  a;
(%o12)  10

(%i21)  f;
(%o21)  f

(%i22)  f=5;
(%o22)  f=5

(%i23)  f;
(%o23)  f

(%i24)  eqn1: _a0*x-5*y=17;
(eqn1)  _a0 x-5 y=17

(%i25)  eqn2: _a1*x+3+y=29;
(eqn2)  y+_a1 x+3=29

(%i26)  eqn1;
(%o26)  _a0 x-5 y=17

(%i27)  eqn2;
(%o27)  y+_a1 x+3=29
```

A function definition (: =)

For a single variable `x`

`f(x_1, …, x_n) := expr` defines a function named `f` with arguments `x_1, …, x_n` and function body `expr`.
The function body is evaluated every time the function is called.

```
(%i28)  expr: (x^3-1)^2;
(expr)  (x^3-1)^2

(%i29)  f(x):=expr;
(%o29)  f(x):=expr

(%i30)  f(x);
(%o30)  (x^3-1)^2
```

A function definition (: =)

For a multiple variables `y` and `z`

`f(x_1, …, x_n) := expr` defines a function named `f` with arguments `x_1, …, x_n` and function body `expr`.
The function body is evaluated every time the function is called.

```
(%i1) expr : cos(y) - sin(x);
(%o1)                        cos(y) - sin(x)
(%i2) F1 (x, y) := expr;
(%o2)                        F1(x, y) := expr
```

Agenda

- Introduction to Maxima
- **Maxima operators**
  - Input and output
  - Starting function `kill()`
  - Terminator and special characters
  - Assignment (`:`)
  - Equation (=)
  - Function (`:=`)
  - Declarations `assume()` and `define()`
  - Numerical evaluation `numer` and `ev()`
  - Arithmetic operation and trigonometric functions
  - Differentiation and integrals `diff()` and `integrate()`
  - Polynomials
  - System of equations `solve()` and `linsolve()`
  - Two-dimensional plot `plot2d()`
- References
- Castigliano’s Theorem
- References

Declarations

assume()

Operators	Symbol
Adds predicates `pred_1, …, pred_n` to the current context. This declaration returns a list whose elements are the predicates added to the context. The predicates `pred_1, …, pred_r` can only be expressed with the relational operators `<` `>=` `equal` `notequal` `>=` and `>`. Predicates cannot be literal equality `=` or literal inequality `!=`.	`assume(pred_1, …., pred_n);`

```
-->      assume (xx>0, yy<-1, zz>=0);
(%o31)   [xx>0,yy<-1,zz>=0]

(%i32)   assume (equal (ww,0), notequal (qq,1));
(%o32)   [equal(ww,0),notequal(qq,1)]
```

Declarations

`define()`

Operators	Symbol
Defines a function named `f` with arguments `x_1, …, x_n` and function body `expr`. `Define` always evaluates its second argument. The function so defined may be an ordinary Maxima function (with argument enclosed in the parentheses) or an array function (with arguments enclosed in squared brackets). When the first argument of `define` is an expression on the form `f(x_1, …, x_n)` or `f[x_1, …, x_n]`, the function arguments are evaluated but `f` is NOT evaluated, even if there is already a function or a variable by that name.	`define(f(x_1, …, x_n), expr);` `define(f[x_1, …, x_n], expr);`

Examples

define () vs :=

```
(%i1) expr : cos(y) - sin(x);
(%o1)                        cos(y) - sin(x)
(%i2) define (F1 (x, y), expr);
(%o2)                  F1(x, y) := cos(y) - sin(x)
(%i3) F1 (a, b);
(%o3)                        cos(b) - sin(a)
(%i4) F2 (x, y) := expr;
(%o4)                       F2(x, y) := expr
(%i5) F2 (a, b);
(%o5)                        cos(y) - sin(x)
```

Numerical evaluation

`numer, ev()`

Operators	Symbol
Allow the numerical evaluation of an expression in floating point. `Numer` causes some mathematical function (including exponentiation) with numerical arguments to be evaluated in floating point. It causes variables in expr which have been given numervals to be replaced by their values.	`%, numer;`
Evaluates the expression expr in the enviroment specified by the arguments `arg_1, … arg_r`. The operator `ev` returns the results (another expression) of the evaluation.	`ev(expr, arg_1, …, arg_r);`

Examples

`ev()` and `%,numer`

```
(%i28)  expr: (x^3-1)^2;
```
(expr) $\left(x^3-1\right)^2$

```
(%i29)  f(x):=expr;
```
(%o29) $f\left(x\right) := expr$

```
(%i30)  f(x);
```
(%o30) $\left(x^3-1\right)^2$

```
(%i38)  expr;
```
(%o38) $\left(x^3-1\right)^2$

```
(%i39)  %, numer, x=3/2;
(%o39)  5.640625
```

```
(%i40)  ev(expr,x=3/2);
```
(%o40) $\frac{361}{64}$

```
(%i52)  ev(expr,x=3/2), numer;
(%o52)  5.640625
```

Maxima

Arithmetic operations and trigonometric functions

Operators	Symbol
Addition	`+`
Subtraction	`-`
Scalar Multiplication	`*`
Division	`/`
Exponentiation	`^` or `**`
Matrix multiplication	`.`
Square root of `x` variable	`sqrt(x)`
Funzione seno	`sin(x);`
Funzione coseno	`cos(x);`

Agenda

- Introduction to Maxima
- **Maxima operators**
  - Input and output
  - Starting function `kill()`
  - Terminator and special characters
  - Assignment (`:`)
  - Equation (=)
  - Function (`:=`)
  - Declarations `assume()` and `define()`
  - Numerical evaluation numer and `ev()`
  - Arithmetic operation and trigonometric functions
  - Differentiation and integrals `diff()` and `integrate()`
  - Polynomials
  - System of equations `solve()` and `linsolve()`
  - Two-dimensional plot `plot2d()`
- References
- Castigliano's Theorem
- References

Maxima

Differentiation and integrals

Operators	Symbol
Differentiation `n`-th order	`diff(expr, x, n);`
Differentiation of first order Implicit assumption that `n` is equal to `1`.	`diff(expr, x);`
Indefined integral	`integrate(expr, x);`
Defined integral	`integrate(expr, x, a, b);`

Maxima

`diff()`

Operators	Symbol
Differentiation `n`-th order. Returns the `n`-th derivative of `expr` with respect to variable `x`.	`diff(expr, x, n);`
Differentiation of first order Implicit assumption that `n` is equal to `1`. Returns the first derivative of `expr` with respect to variable `x`.	`diff(expr, x);`

Maxima

`integrate()`

Operators	Symbol
Indefined integral Attempts to symbolically compute the integral of `expr` with respect to `x`.	`integrate(expr, x);`
Defined integral The defined integral has the limits of integration called `a` and `b`. The limits should not contain `x`, although integrate does not enforce this restriction. `a` need not be lower than `b`. If `b` is equal to `a`, integrate returns zero.	`integrate(expr, x, a, b);`

Examples

`diff()` and `integrate()`

```
(%i24)  expr: (x^3-1)^2;
(expr)
```
$$\left(x^3-1\right)^2$$

```
(%i25)  f(x):=expr;
(%o25)
```
$$f\left(x\right):=expr$$

```
(%i26)  f(x);
(%o26)
```
$$\left(x^3-1\right)^2$$

Differential

```
(%i27)  diff(f(x),x,1);
(%o27)
```
$$6\,x^2\left(x^3-1\right)$$

```
(%i28)  diff_first:%;
(diff_first)
```
$$6\,x^2\left(x^3-1\right)$$

```
(%i29)  diff(f(x),x,2);
(%o29)
```
$$18\,x^4+12\,x\left(x^3-1\right)$$

```
(%i30)  diff_sec:%;
(diff_sec)
```
$$18\,x^4+12\,x\left(x^3-1\right)$$

Integral

```
(%i31)  integrate(f(x), x);
(%o31)
```
$$\frac{x^7}{7}-\frac{x^4}{2}+x$$

```
(%i32)  integrate(f(x), x, -2, 1);
(%o32)
```
$$\frac{405}{14}$$

Polynomials
Factorization, simplification and expansion

Operators	Symbol
Factors the expression expr, containing any number of variables or functions, into factors irreducibe over the integer.	`factor(expr);`
Simplifies the expression expr and all of its subexpressions, including the arguments to non-rational functions.	`fullratsimp(expr);`
Product of sums and exponentiated sums are multiplied out.	`expand(expr);`

# Examples
factor() fullratsimp() expand()

```
(%i33)   expr;
```
(%o33) $\left(x^3-1\right)^2$

```
(%i34)   factor(expr);
```
(%o34) $\left(x-1\right)^2\left(x^2+x+1\right)^2$

```
(%i35)   fullratsimp(expr);
```
(%o35) $x^6-2\,x^3+1$

```
(%i36)   expr2:(g+h)^5;
```
(expr2) $\left(h+g\right)^5$

```
(%i37)   expand(expr2);
```
(%o37) $h^5+5\,g\,h^4+10\,g^2\,h^3+10\,g^3\,h^2+5\,g^4\,h+g^5$

System of equations
Numerical method

Operators	Symbol
Solves the algebraic equation `expr` for the variable `x` and returns a list of solution equations in `x`. If `expr` is not an equation, the equation `expr=0` is assumed in its place. `x` may be omitted if the `expr` contains only one variable.	`solve(expr, x);` `solve(expr);`
Solves a system of simultaneous (linear or non-linear) polynomial equations, and returns a list of solutions lists in the variables.	`solve([eqn_1, …, eqn_n], [x_1, …, x_n]);`
Solves a system of simultaneous LINEAR polynomial equations, and returns a list of solutions lists in the variables.	`linsolve([expr_1, …, expr_n], [x_1, …, x_n];`

System of equations

Numerical method: options

Operators	Symbol
Each solved-for variable is bound to its value in the solution of the equations.	`globalsolve = true;`

Example

`linsolve()` and `globalsolve=true`

```
(%i53)   eqn_1: x+z=y;
(eqn_1)  z+x=y

(%i54)   eqn_2:2*t*x-y=2*t^2;
(eqn_2)  2 t x-y=2 t^2

(%i55)   eqn_3:y-2*z=2;
(eqn_3)  y-2 z=2

(%i56)   linsolve([eqn_1, eqn_2, eqn_3], [x,y,z]);
(%o56)   [x=t+1,y=2 t,z=t-1]

(%i57)   x;
(%o57)   x

(%i58)   linsolve([eqn_1, eqn_2, eqn_3], [x,y,z]), globalsolve=true;
(%o58)   [x:t+1,y:2 t,z:t-1]

(%i59)   x;
(%o59)   t+1
```

Example

`linsolve()` vs `solve()`

```
(%i60)  kill(x,y);
(%o60)  done
(%i61)  linsolve([x+3*y=2, 2*x-y=5],[x, y]);
```

(%o61) $[x=\frac{17}{7}, y=-\frac{1}{7}]$

```
(%i62)  kill(eqn, x, y);
(%o62)  done
(%i63)  eqn:[4*x^2-y^2=12,x*y-x=2];
```

(eqn) $[4x^2-y^2=12, xy-x=2]$

```
(%i64)  solve(eqn, [x,y]);
(%o64)  [[x=2,y=2],[x=0.5202594388652008 %i-0.1331240357358706,y=0.07678378523787788-
3.608003221870287 %i],[x=-0.5202594388652008 %i-0.1331240357358706,y=3.608003221870287 %i+
0.07678378523787788],[x=-1.733751846381093,y=-0.1535675710019696]]
(%i65)  %[2];
(%o65)  [x=0.5202594388652008 %i-0.1331240357358706,y=0.07678378523787788-3.608003221870287 %i]
```

Example

solve()

(%i66) kill(a, x);

(%o66) done

(%i67) solve(1+a*x+x^3, x);

(%o67) $$[x=\left(-\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}-\frac{\left(\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)a}{3\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}},x=\left(\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}-\frac{\left(-\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)a}{3\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}},x=\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}-\frac{a}{3\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}}]$$

(%i68) %[1];

(%o68) $$x=\left(-\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}-\frac{\left(\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)a}{3\left(\frac{\sqrt{4\,a^3+27}}{2\,3^{3/2}}-\frac{1}{2}\right)^{1/3}}$$

Agenda

- Introduction to Maxima
- **Maxima operators**
  - Input and output
  - Starting function `kill()`
  - Terminator and special characters
  - Assignment (`:`)
  - Equation (`=`)
  - Function (`:=`)
  - Declarations `assume()` and `define()`
  - Numerical evaluation `numer` and `ev()`
  - Arithmetic operation and trigonometric functions
  - Differentiation and integrals `diff()` and `integrate()`
  - Polynomials
  - System of equations `solve()` and `linsolve()`
  - Two-dimensional plot `plot2d()`
- References
- Castigliano's Theorem
- References

Plotting

two-dimensions plot

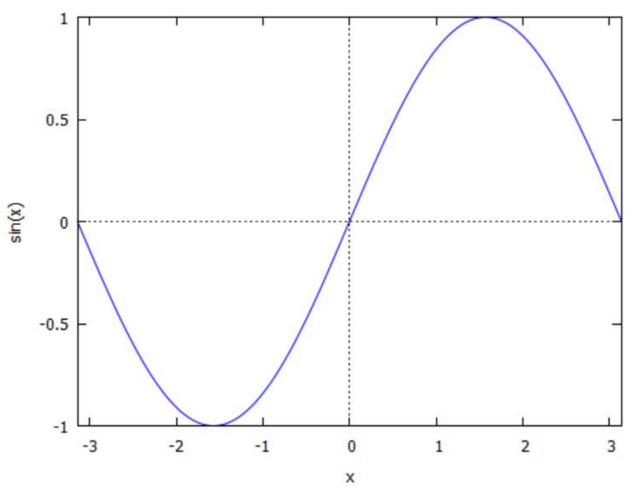
Operators	Symbol

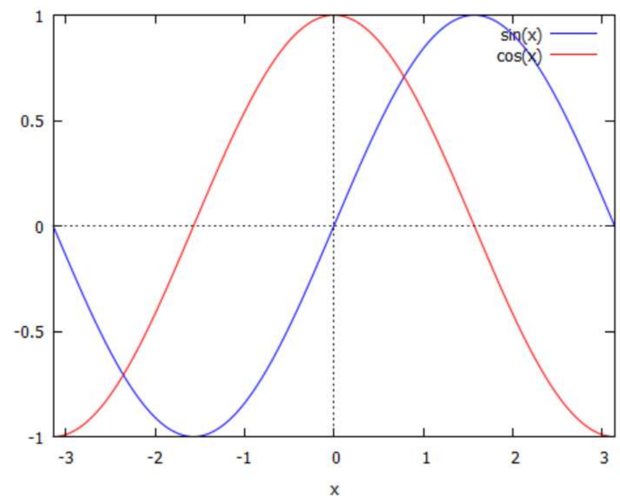
Displays one or several plots in two dimensions. When expressions or function name are used to define the plots, they should all depend on only one variable `var` and the use of `x_range` will be mandatory, to provide the name of the variable and its minimum and maximum values.
The syntax for the `x_range` is `[variable, min, max]`
	`plot2d(plot, x_range, …, options);`
	
	`plot2d([plot_1, …, plot_n], …, options);`

Example

plot2d()

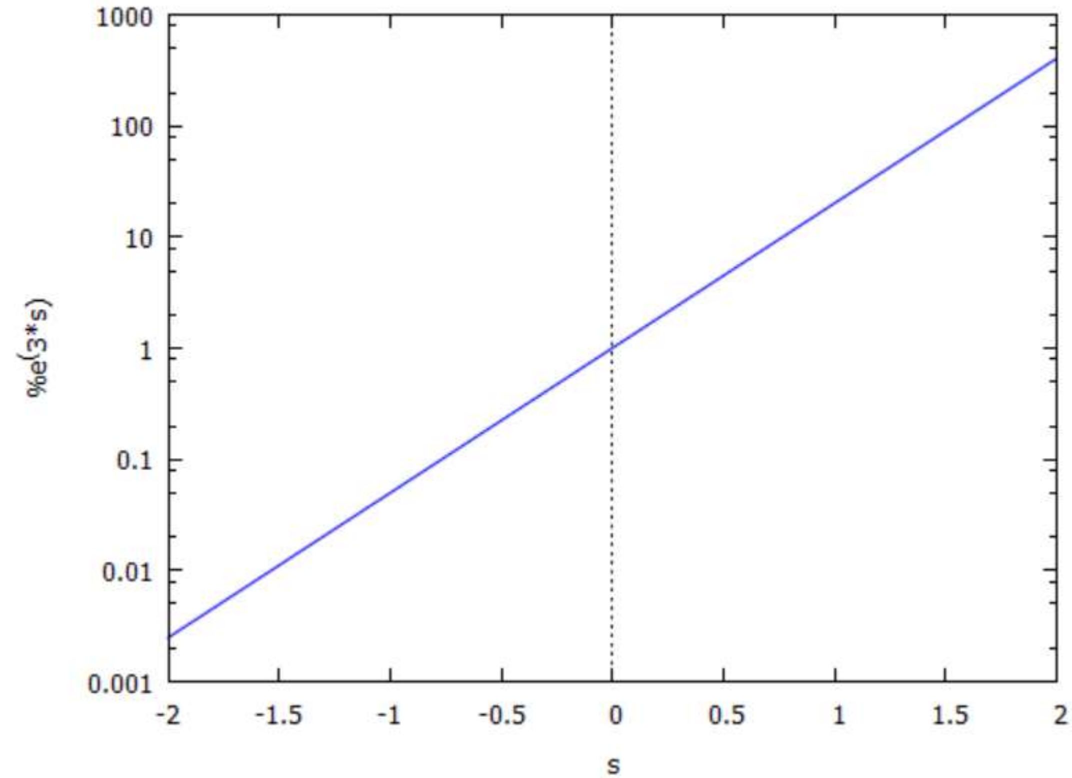
```
(%i69)  kill(all);
(%o0)   done

(%i1)   plot2d ([sin(x)], [x, -%pi, %pi]);
(%o1)   [C:/Users/manto/maxout12624.gnuplot]

(%i2)   plot2d ([sin(x), cos(x)], [x, -%pi, %pi])$
```





Example

plot2d()




```
(%i69)  kill(all);
(%o0)   done

(%i1)   plot2d ([sin(x)], [x, -%pi, %pi]);
(%o1)   [C:/Users/manto/maxout12624.gnuplot]

(%i2)   plot2d ([sin(x), cos(x)], [x, -%pi, %pi])$

(%i3)   plot2d(%e^(3*s), [s, -2, 2], logy)$
```

Agenda

- Introduction to Maxima
- Maxima operators
- **References**
- Castigliano's Theorem
- References

References

ENGLISH DOCUMENTATION

- http://maxima.sourceforge.net/docs/manual/maxima.html
- http://maxima.sourceforge.net/docs/tutorial/en/minimal-maxima.pdf (miniguide)

- http://superk.physics.sunysb.edu/~mcgrew/phy310/documentation/maxima-reference.pdf (extensive guide)

ITALIAN DOCUMENTATION

http://maxima.sourceforge.net/docs/tutorial/it/maxima_1.0-consonni.pdf (miniguide)

LAB Maxima file saved as

`intro_maxima_operators.wxmx`

Agenda

- Introduction to Maxima
- Maxima operators
- References
- **Castigliano's Theorem**
  - Theorem of least work
  - CASE A: Statically determined structure
  - CASE B: Statically determined structure
  - CASE C: Statically redundant structure (+1 dof)
  - CASE D: rollbar- statically redundant structure (+3 dof)
- References

Castigliano's Theorem

Theorem of least work

"The first partial derivative of the total internal energy (U) in a structure with respect to the (force P) (couple C) applied at any point is equal to the (deflection δ*)* (angular rotation φ) *at the point of application of that (force) in the direction of its line of action (or couple)".*

$$\delta = \frac{\partial U}{\partial P};$$

$$\varphi = \frac{\partial U}{\partial C}$$

The theorem is applicable to linearly elastic (Hookean material) structures with constant temperature and unyielding supports.

Note that in the above statements:

- *force (P)* may mean point force or *couple (C);*
- *displacement* may mean translation (δ) or angular rotation (φ).

Castigliano's Theorem
Theorem of least work

The Castigliano's theorem:

1- applied to a **statically determined structure,** allows the deflection and the angular rotation of the structure to be computed;

2- applied to a **statically redundant structure,** allows the reaction forces to be determined. Therefore, the structure becomes a statically determined structure consequently the deflection and the angular rotation of the redundant structure are also computed.

Castigliano's Theorem
Theorem of least work

Considering only **plane** problems, the internal energy (*U*) of the structure is:

$$U = \int_l \left(\frac{M_f^2}{2EJ} + \frac{N^2}{2AE} + \xi \frac{T^2}{2AG} \right) dx$$

where:
l is the length of the structure;
M_f, N, T are the bending moment, normal force and the shear force;
A and *J* are the cross-section area and moment of inertia;
E and *G* are the Young's modulus and shear modulus of the material;
ξ Is the shear coefficient associated to the evaluation the internal energy done by the shear force, where ξ is 1.2 or 1.11 for rectangular or circular cross-section beam.

Castigliano's Theorem

CASE A: Statically determined structure

Considering a cantilever beam in steel loaded by a concentrated force P at the extremity called point A and fixed to the further extremity, evaluate the deflection of the beam at the point A (δ_A).

Hp) Rectangular cross section
b: 10 mm, h:20 mm;
l: 100 mm
P: 10000 N

$$M_f(x) = Px \quad ; \quad T = P$$

$$U = \int_0^l \frac{M_f^{\,2}}{2EJ}dx + \int_0^l \xi \frac{T^2}{2AG}dx = \int_0^l \frac{P^2x^2}{2EJ}dx + \int_0^l \xi \frac{P^2}{2AG}dx = \frac{P^2 l^3}{2\times 3EJ} + \xi \frac{P^2 l}{2AG}$$

$$\delta_A = \frac{\partial U}{\partial P} = \frac{Pl^3}{3EJ} + \xi \frac{Pl}{AG}$$

Castigliano’s Theorem

CASE A: Statically determined structure

Rectangular cross section
b: 10 mm, h:20 mm;
l: 100 mm
P: 10000 N
ξ :1.2
J: $b*h^3/12$: 6666,66 mm^4
A: b*h: 200 mm^2
E: 210000 MPa
ν: 0,3
G: $E/[2(1+\nu)]$: 80000 MPa (ca)

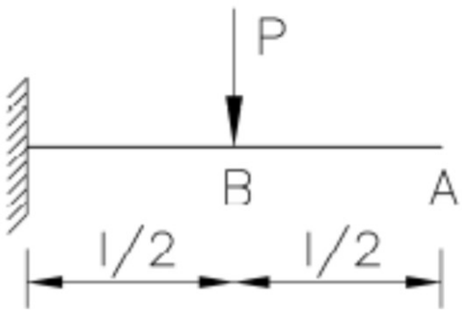
$$\delta_A = \frac{\partial U}{\partial P} = \frac{Pl^3}{3EJ} + \xi \frac{Pl}{AG}$$

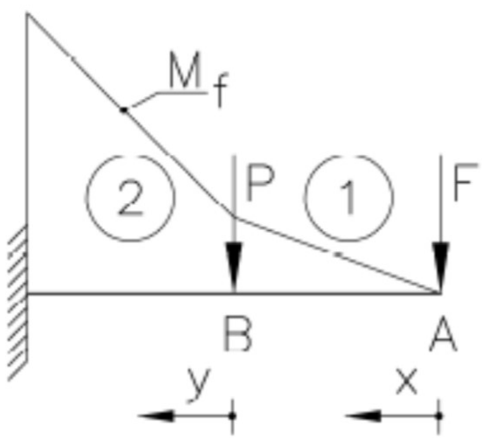
δ_A = 2,3810 + 0.075 = 2,456 mm

The shear contributes to the deflection of the beam the 3.05 per cent, due to this limited contribution the shear is commonly omitted from the preliminary dimensioning of a structure.

Castigliano's Theorem

CASE B: Statically determined structure





Considering a cantilever beam loaded by a concentrated force P at the midspan of the beam (point B) and fixed to one extremity, evaluate the deflection of the beam at the free extremity of the beam at the point A (δ_A).

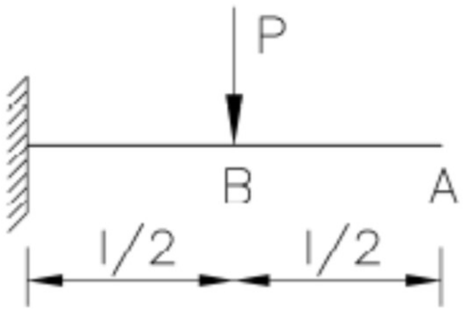
NOTE: The point A is an unloaded section of the beam.

*"The first partial derivative of the total internal energy (U) in a structure with respect to the force P applied at any point is equal to the deflection δ at the point of application of **that force** in the direction of its line of action".*

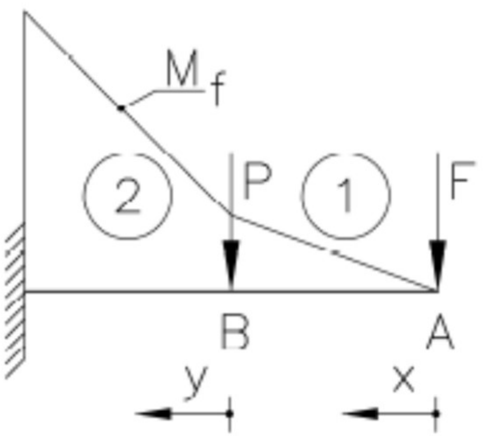
Hp) the shear contribution has been neglected from the internal energy equation.

Castigliano's Theorem

CASE B: Statically determined structure



A fictitious concentrated force F is applied to the beam, at the point A; at which the deflection of the beam must be evaluate.
At the end of the calculus that force F will be considered as null.



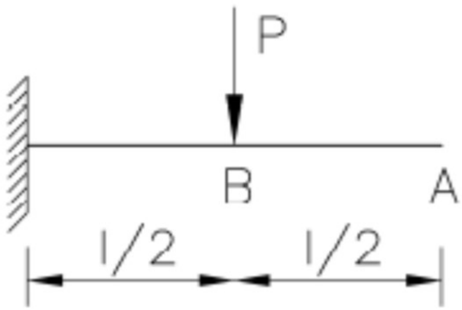
Part (1)

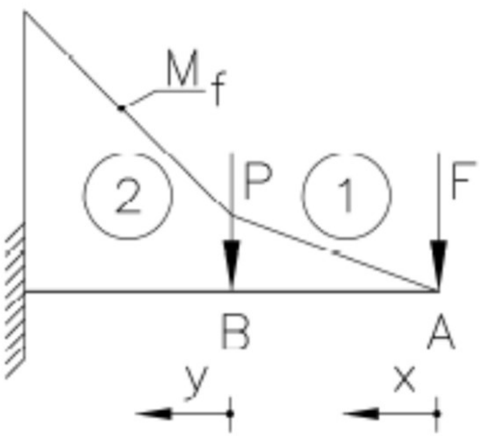
$$M_{f,1}(x) = Fx$$

$$U_1 = \int_0^{l/2} \frac{M_{f,1}^{\ 2}}{2EJ} dx = \int_0^{l/2} \frac{(Fx)^2}{2EJ} dx = \frac{F^2}{2EJ} \int_0^{l/2} x^2 dx = \frac{F^2}{2EJ} \frac{x^3}{3} \Big|_0^{l/2} = \frac{F^2}{2EJ} \frac{l^3}{3 \times 8} = \frac{F^2 l^3}{48EJ}$$

Castigliano’s Theorem

CASE B: Statically determined structure





Part (2)

$$M_{f,2}(x) = Py + F(y + \frac{l}{2})$$

$$U_2 = \int_0^{l/2} \frac{M_{f,2}^{\ 2}}{2EJ} dy = \int_0^{l/2} \frac{\left(Py + F(y+\frac{l}{2})\right)^2}{2EJ} dy =$$

$$\frac{1}{2EJ}\int_0^{l/2}\left(P^2y^2 + F^2y^2 + F^2\frac{l^2}{4} + 2F^2y\frac{l}{2} + 2PFy^2 + 2PFy\frac{l}{2}\right)dy =$$

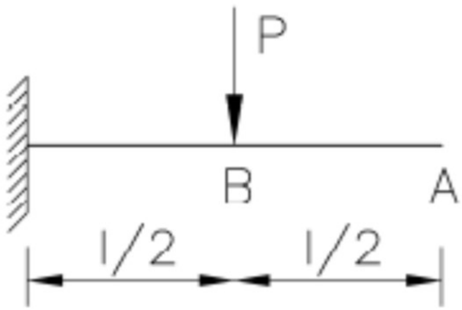
$$\frac{1}{2EJ}\left(P^2\frac{l^3}{3\times 8} + F^2\frac{l^3}{3\times 8} + F^2\frac{l^2}{4}\frac{l}{2} + 2F^2\frac{l}{2}\frac{l^2}{2\times 4} + 2PF\frac{l^3}{3\times 8} + 2PF\frac{l}{2}\frac{l^2}{2\times 4}\right) =$$

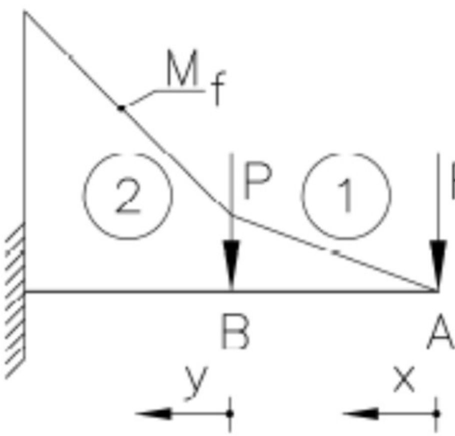
$$\frac{1}{2EJ}\left(P^2\frac{l^3}{24} + F^2\frac{l^3}{24} + F^2\frac{l^3}{8} + F^2\frac{l^3}{8} + PF\frac{l^3}{12} + PF\frac{l^3}{8}\right) =$$

$$\frac{1}{48EJ}\left(P^2l^3 + 7F^2l^3 + 5PFl^3\right)$$

Castigliano's Theorem

CASE B: Statically determined structure





Part (2)

$$M_{f,2}(x) = Py + F(y + \frac{l}{2})$$

$$U_2 = \int_0^{l/2} \frac{M_{f,2}{}^2}{2EJ} dy = \int_0^{l/2} \frac{\left(Py + F(y + \frac{l}{2})\right)^2}{2EJ} dy =$$

$$\frac{1}{2EJ} \int_0^{l/2} \left(P^2 y^2 + F^2 y^2 + F^2 \frac{l^2}{4} + 2F^2 y \frac{l}{2} + 2PFy^2 + 2PFy \frac{l}{2} \right) dy =$$

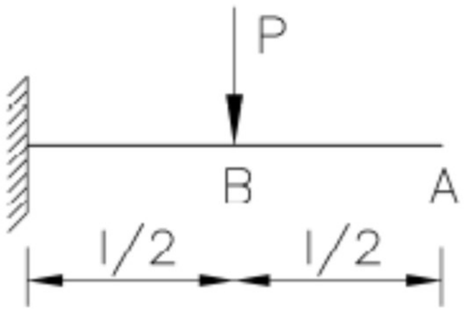
$$\frac{1}{2EJ} \left(P^2 \frac{l^3}{3\times 8} + F^2 \frac{l^3}{3\times 8} + F^2 \frac{l^2}{4}\frac{l}{2} + 2F^2 \frac{l}{2}\frac{l^2}{2\times 4} + 2PF \frac{l^3}{3\times 8} + 2PF \frac{l}{2}\frac{l^2}{2\times 4} \right) =$$

$$\frac{1}{2EJ} \left(P^2 \frac{l^3}{24} + F^2 \frac{l^3}{24} + F^2 \frac{l^3}{8} + F^2 \frac{l^3}{8} + PF \frac{l^3}{12} + PF \frac{l^3}{8} \right) =$$

$$\frac{1}{48EJ} \left(P^2 l^3 + 7F^2 l^3 + 5PFl^3 \right)$$

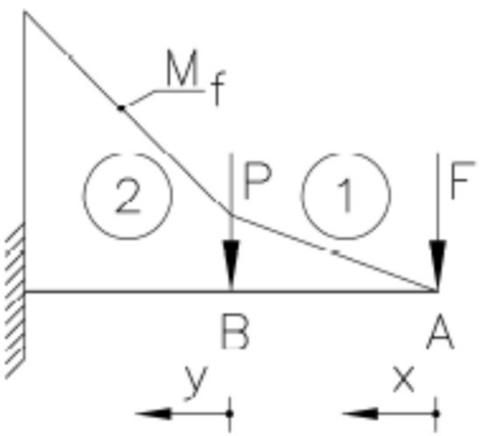
Castigliano's Theorem

CASE B: Statically determined structure



Total internal energy of the structure (U)

$$U = U_1 + U_2 = \frac{F^2 l^3}{48EJ} + \frac{1}{48EJ}\left(P^2 l^3 + 7F^2 l^3 + 5PFl^3\right)$$
$$= \frac{1}{48EJ}\left(P^2 l^3 + 8F^2 l^3 + 5PFl^3\right)$$



The fictitious concentrated force F is assume null, at the conclusion of the calculus. *LAST EVALUATION!!!*

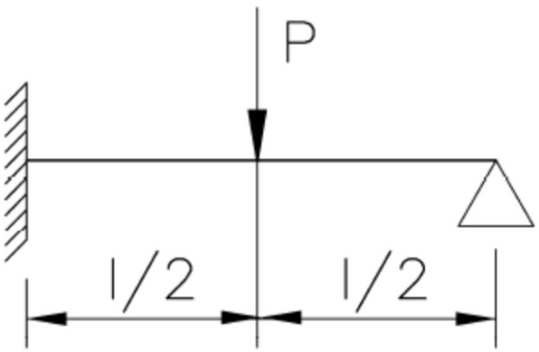
$$\delta_A = \left.\frac{\partial U}{\partial F}\right|_{F=0} = \frac{1}{48EJ}\left.\left(16Fl^3 + 5Pl^3\right)\right|_{F=0}$$
$$= \frac{1}{48EJ}5Pl^3 = \frac{5Pl^3}{48EJ}$$

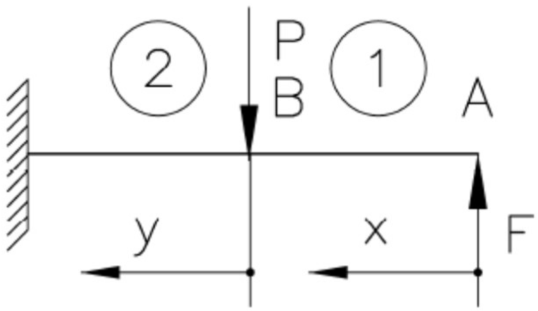
Agenda

- Introduction to Maxima
- Maxima operators
- References
- **Castigliano's Theorem**
  - Theorem of least work
  - CASE A: Statically determined structure
  - CASE B: Statically determined structure
  - CASE C: Statically redundant structure (+1 dof)
  - CASE D: Rollbar- statically redundant structure (+3 dof)
- References

Castigliano's Theorem

CASE C: Statically redundant structure (+1dof)





Considering a beam:

- loaded by a concentrated force P at the midspan of the beam (point B);
- fixed to right-hand side extremity;
- simply-supported to the left-hand side extremity at the point A

Evaluate the reaction force (*F*) acting at the support at point A of the structure.

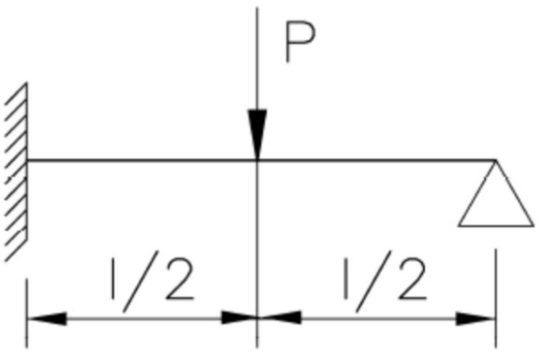
The equilibrium equations are not sufficient to calculate the reaction forces acting on the beam.

Therefore, a compatibility equation related to the deformation of the structure must be imposed.

In this case, the A-point of the beam does NOT be able to move vertically.

Castigliano's Theorem

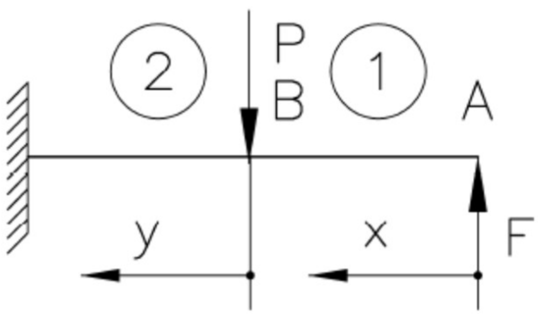
CASE C: Statically redundant structure (+1dof)



Part (1)

$$M_{f,1}(x) = Fx$$

$$U_1 = \int_0^{l/2} \frac{{M_{f,1}}^2}{2EJ}dx = \int_0^{l/2} \frac{F^2x^2}{2EJ}dx = \frac{F^2}{2EJ}\frac{l^3}{3\times 8}$$



Part (2)

$$M_{f,2}(y) = F\left(y + \frac{l}{2}\right) - Py$$

$$U_2 = \int_0^{l/2} \frac{{M_{f,2}}^2}{2EJ}dx =$$

$$\frac{1}{2EJ}\left[F^2\left(\frac{l^3}{3\times 8} + \frac{l^2}{4}\frac{l}{2} + 2\frac{l}{2}\frac{l^2}{2\times 4}\right) + P^2\frac{l^3}{3\times 8} - 2PF\left(\frac{l^3}{3\times 8} + \frac{l}{2}\frac{l^2}{2\times 4}\right)\right]$$

Castigliano's Theorem

CASE C: Statically redundant structure (+1dof)

Total internal energy of the structure (*U*)

$$U = U_1 + U_2 = \frac{F^2}{2EJ}\frac{l^3}{3\times 8} + \frac{1}{2EJ}\left[F^2\left(\frac{l^3}{3\times 8} + \frac{l^2}{4}\frac{l}{2} + 2\frac{l}{2}\frac{l^2}{2\times 4}\right) + P^2\frac{l^3}{3\times 8} - 2PF\left(\frac{l^3}{3\times 8} + \frac{l}{2}\frac{l^2}{2\times 4}\right)\right]$$

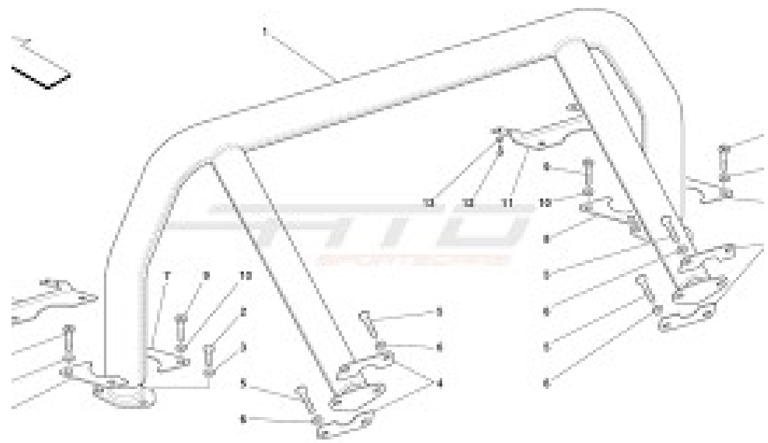
Evaluate the deflection of the beam at the point A (δ_A) by the first partial derivative of the total internal energy (*U*) proper of the structure with respect to the force *F*.
Imposing that the deflection (δ_A) is forbidden due to the presence of the support acting at the point A, the unknown of the problem is the vertical reaction force (equal to *F*) that the support induced into the structure.

$$\delta_A = \frac{\partial U}{\partial F} = \frac{2F}{2EJ}\frac{l^3}{3\times 8} + \frac{1}{2EJ}\left[2F\left(\frac{l^3}{3\times 8} + \frac{l^2}{4}\frac{l}{2} + 2\frac{l}{2}\frac{l^2}{2\times 4}\right) - 2P\left(\frac{l^3}{3\times 8} + \frac{l}{2}\frac{l^2}{2\times 4}\right)\right] =$$

$$\frac{1}{2EJ}\left[\frac{2Fl^3}{3} - \frac{5Pl^3}{24}\right] = 0 \quad \Rightarrow \quad F = \frac{5P}{16}$$

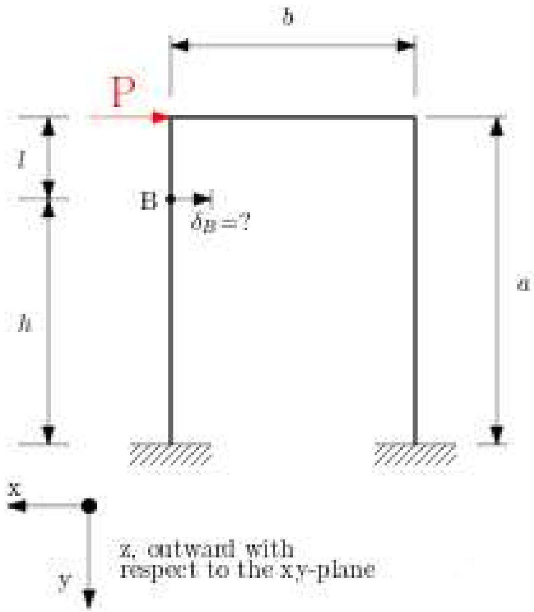
Castigliano's Theorem

CASE D: Roll bar - Statically redundant structure (+3 dof)



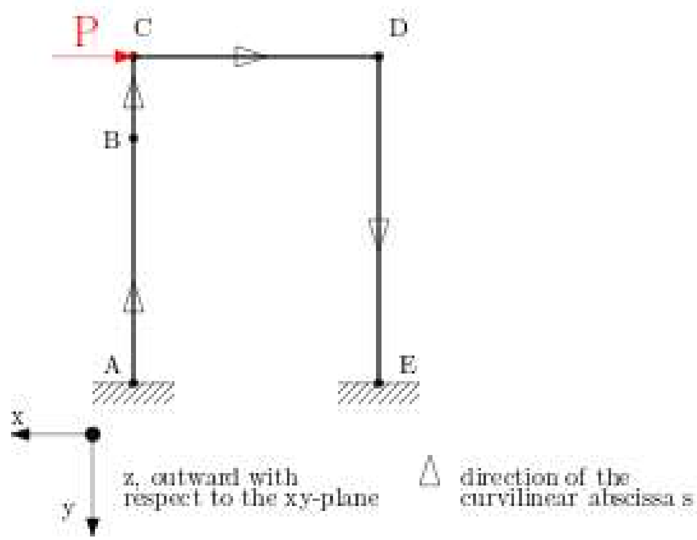
Castigliano's Theorem

CASE D: Roll bar - Statically redundant structure (+3 dof)



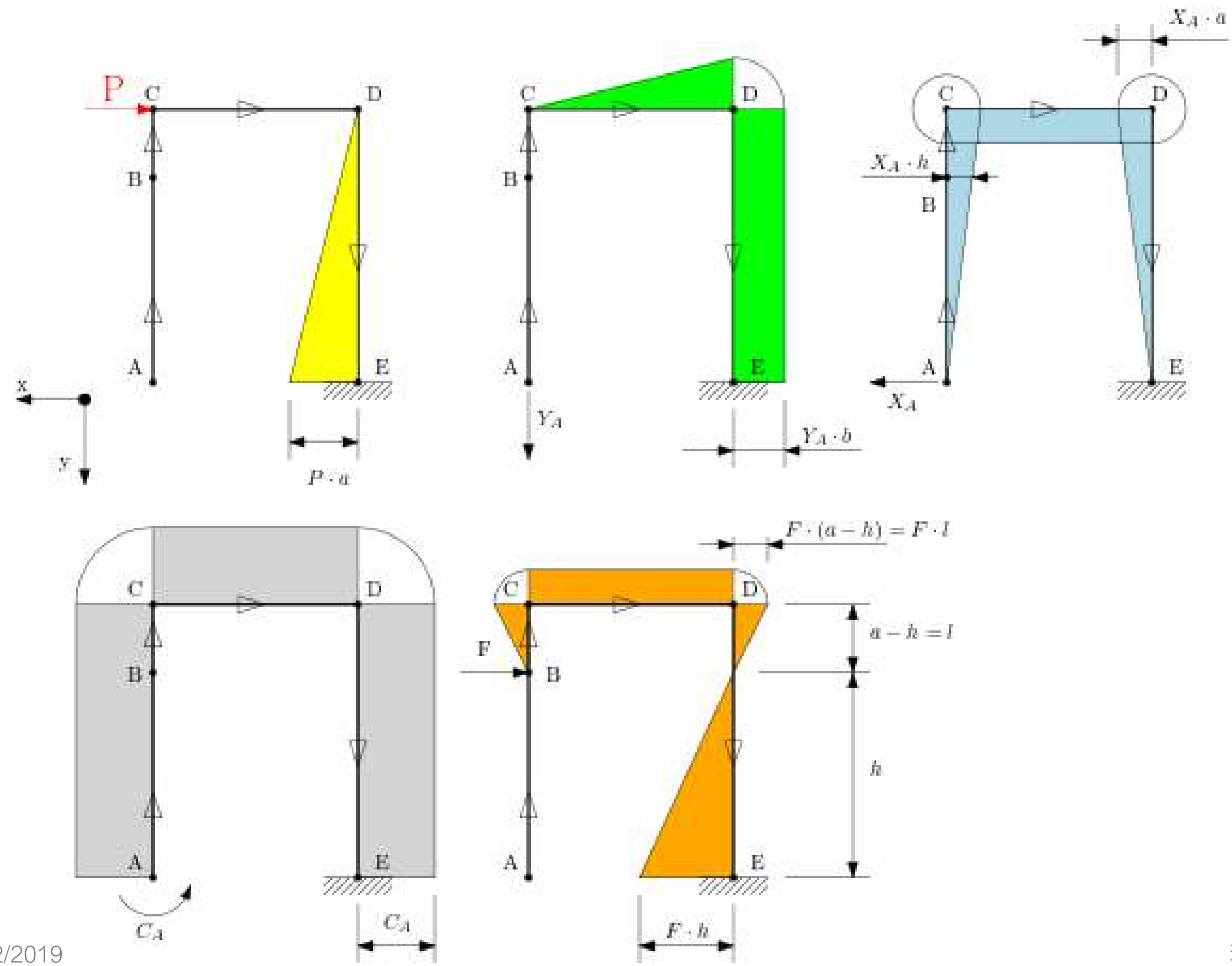
Considering a simplified roll bar:
- fixed to the extremities;
- loaded by a lateral concentrated force (P) acting at the point B of the structure.

Evaluate the deflection (δ_B) acting at the point B of the structure, located at the maximum point at which the driver and the passenger can reach during a rollover crash event.



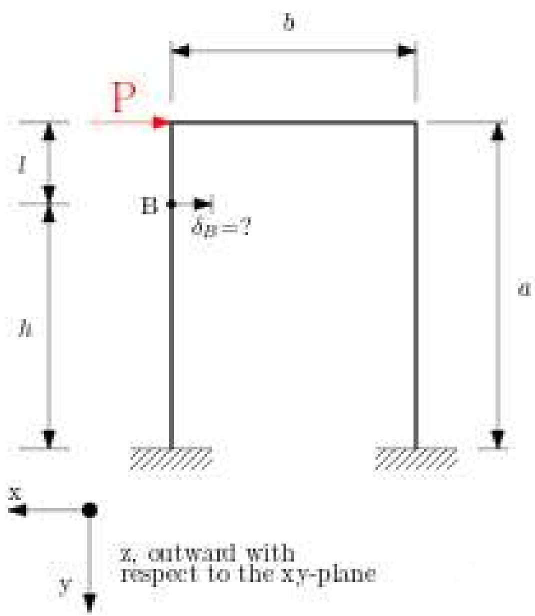
Castigliano’s Theorem

CASE D: Roll bar - Statically redundant structure (+3 dof)



Castigliano's Theorem

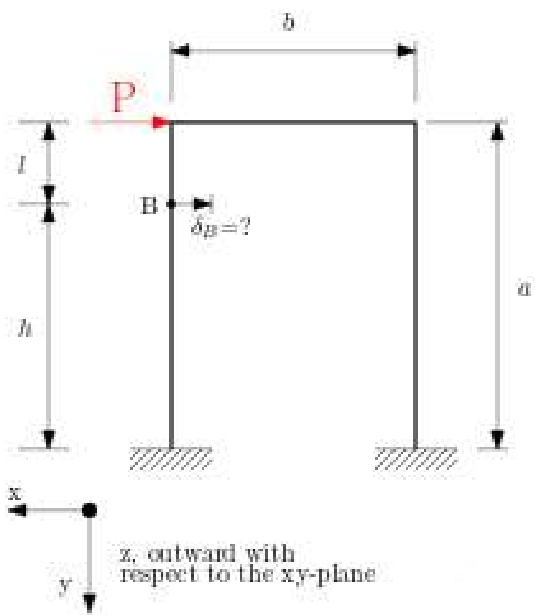
CASE D: Roll bar - Statically redundant structure (+3 dof)



1. Application of a fictitious force *F* at point B;
2. Evaluation of the equilibrium of the structure, to retrieve the reaction forces and moment acting at E;
3. Definition of a linear shape functions in the [0,1] interval;
4. Definition of the bending moment acting on the portions of the structure, called as `Mf_AB, Mf_BC, Mf_CD, Mf_DE`;
5. Definition of the elastic internal energy related to the various beam segments `U_AB, U_BC, U_CD, U_DE`;
6. Evaluation of the total elatic internal energy of the structure defined as the sum of the various beam segments.

Castigliano's Theorem

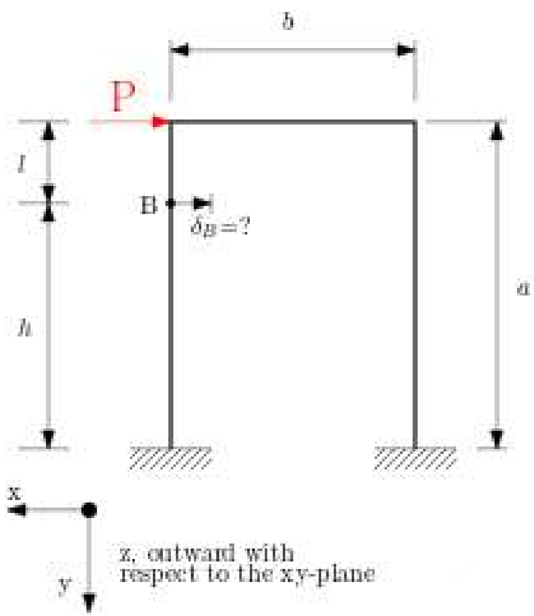
CASE D: Roll bar - Statically redundant structure (+3 dof)



7. Application of the Castigliano's theorem for obtaining the displacements and the rotation at A named as `uA, vA, rA.`
8. Definition of kinematic congruence with respect to the clamp constraint in A is to be enforced, by the mean of a system of (linear) equations.
9. Evaluation of the redundant reaction force and moment acting at point A (`XA, YA, CA`) starting from the system of equation imposed by the kinematic congruence equations (see point 8).

Castigliano’s Theorem

CASE D: Roll bar - Statically redundant structure (+3 dof)



10. Evaluation of the overall internal energy of the structure *U*, substituting the definition of `XA, YA, CA`. The U relation is function of the external load *P* and of the fictitious force *F* acting at B.
11. The displacement at the B point is evaluated through the Castigliano’s theorem;
12. The fictitious nature of *F* may now be enforced to be null.

Q.E.D.

Agenda

- Introduction to Maxima
- Maxima operators
- References
- Castigliano’s Theorem
- **References**

References

ENGLISH DOCUMENTATION

Shigley, J. E. (2011). Shigley's mechanical engineering design. Tata McGraw-Hill Education. Section 4.8 pp. 164-169.

https://eclass.teicrete.gr/modules/document/file.php/TM114/shigley-machine-design-.pdf

ITALIAN DOCUMENTATION

Strozzi, A. (1998). Costruzione di Macchine, Pitagora Ed., Bologna.

Strozzi, A. (2016). Fondamenti di Costruzione di Macchine, Pitagora Ed., Bologna.

LAB Maxima file saved as

`rollbar_def.wxmx`

“Machines can never think as humans do but just because something thinks differently from you, does it mean it's not thinking?”

A. Turing

Sara Mantovani
Via Vivarelli, 10
41125, Modena, Italy
Mail: sara.mantovani@unimore.it
Mail: millechili@unimore.it
Phone: +39 059 2056280