Mathematical Computing

IMT2b2β/MSP3b9β

Department of Mathematics
University of Ruhuna

A.W.L. Pubudu Thilan

Method of evaluation

- Class test
- Project report
- Viva

Class test

- Practical test using Maxima software package
- It will be conducted on Linux platform

Evaluation of project report

- Report structure
- Understanding methodology and use of mathematics
- Use of programing language (Maxima)
- Interpretation of solution(s)
- Discussion/Conclusion

Evaluation of viva

- Mathematics knowledge
- Programing knowledge
- Presentation skills
- Answering questions

Note: All your work should be in your home directory at the time of viva. You are not allowed to use flash drives (pen drives).

Chapter 1

Introduction to the Computer Package Maxima

What is symbolic manipulation?

- It relates to the use of machines, such as computers, to manipulate mathematical equations and expressions in symbolic form.
- Symbolic manipulation is also sometimes referred to as **symbolic computation, symbolic processing, symbolic mathematics,** or **symbolic algebra**.

Examples for some symbolic manipulations

- Simplification to a smaller expression.
- Expanding products and powers.
- Partial and total differentiation.
- Some indefinite and definite integration.
- Solution of linear and some non-linear equations.
- Solution of some differential and difference equations.
- Taking some limits.

Computer Algebra System

- A **Computer Algebra System (CAS)** is a software program that facilitates symbolic manipulations.
- The core functionality of a CAS is manipulation of mathematical expressions in symbolic form.

What is Macsyma?

- **Macsyma** is a CAS that was originally developed from 1968 to 1982 at MIT as part of Project MAC and later marketed commercially.
- It was the first comprehensive symbolic mathematics system and one of the earliest knowledge based systems.
- Many of its ideas were later adopted by **Maxima**, Mathematica, Maple, and other systems.

Development of Maxima

- Maxima is based on a 1982 version of **Macsyma**.
- It is written in **Common Lisp** (dialect of the Lisp programming language).
- Maxima runs on all platforms such as Mac OS X, Unix, BSD, and GNU/Linux as well as under Microsoft Windows.
- Maxima is free software released under the terms of the GNU General Public License (GPL).

More on Maxima

- Maxima is a CAS.
- So, it can be used to manipulate symbolic and numerical expressions.
- Maxima yields high precision numeric results by using exact fractions, arbitrary precision integers, and variable precision floating point numbers.
- Maxima can also plot functions and data in two and three dimensions.

How to run Maxima?

- To run Maxima, type command **maxima** on a terminal.
- You need to use **SHIFT** + **ENTER** to get the line of code to run.
- The semi-colon ';' should be included at the end of each line.
- The semi-colon ends all of the operations you want Maxima to do.

Input and output in Maxima

- The input will be automatically prefixed by %i1.
- The output is prefixed by %o1.
- A command may also be terminated by the special symbol $ instead of a semicolon.
- Then Maxima evaluates your input expression but does not show its results.

Input and output in Maxima

Try followings

(i) 3+4;

(ii) 5*9;

(iii) 2.566*3.45;

(iv) 8-9$

(v) 12/2.3$

(vi) 2.89/23+4$

Input and output in Maxima

More on input

- More than one command can be written on one line.
- On the other hand one command can also be spread over two or more lines.

Input and output in Maxima

More on input⇒Try followings

(i) 4-9; 2*9;

(ii) 32.45/9; 2+9; 6-9*3;

(iii) 4+2*(8+5)
+3*9-6/4;

(iv) 4-9; 2*9$

(v) 4-9$ 2*9$

(vi) 3.5/9+2*(8+5)
+6/9.5-4.78*45$

Maxima Interfaces

- Maxima at its heart has a command line interface and by itself it is not capable of displaying formatted mathematics beyond the plain text level.
- For most users this is unfamiliar and may seem quite difficult.
- Fortunately, nowadays more fancy Graphical User Interfaces (GUIs) are available.
- The most popular one is called **wxMaxima**.
- Alternatives GUIs are **Xmaxima**, **Texmacs** etc.

More on wxMaxima

- **wxMaxima** is a popular cross-platform GUI using wxWidgets.
- wxMaxima provides menus and dialogs for many common Maxima commands, autocompletion, inline plots and simple animations.
- wxMaxima is distributed under the GPL license.

How to run wxMaxima?

- To run wxMaxima, type command **wxmaxima** on a terminal.
- You need to use **SHIFT** + **ENTER** to get the line of code to run.
- It's good practice to include ';' at the end of each line.
- Otherwise wxMaxima automatically added a ';' to the end of your line.
- Pressing **ENTER** alone (without pressing **SHIFT**) only inserts a line break even when a ; or $ is present.

wxMaxima notebook

- wxMaxima provides a more convenient GUI.
- It shows a **notebook** where one can insert an expression.
- The notebook has been organized using cells.
- The bar on the left hand side indicates which input and output cells belong together.
- wxMaxima also provides a toolbar where one can select commands from menus.

Entering text

- Titles, sections, subsections, and text can be included to comment ones calculations.
- For this purpose use the corresponding entries in menu **Cell**.

Saving a notebook

- When a Maxima session is finished one can save a notebook using **File** $\rightarrow$ **Save**.
- Also previous works can be reloaded using **File** $\rightarrow$ **Open**.

Help

- Help pages for all commands and operators are available using the special symbols **? cmd** and **?? cmd**.
- The double question mark **??** can be used to search for string **cmd** in the manual.
- It is important to insert a space between **?** and **cmd**.

Help with command apropos

- If you only remember a substring of a command name, then **apropos** is quite useful.
- It returns a list of all those Maxima names that contain this string.
- By using **apropos** with substring “sqr”, we can find Maxima names which include “sqr”.

  (%i1) apropos("sqr");
  (%o1)[isqrt,sqrt,sqrtdispflag]

Help in menu bar

- For an alternative method to access the manual within wxMaxima press the F1 button or use the Help button in the menu bar.
- It gives access to the whole library including an index and a search function.

Numerical Computations

Arithmetic operations

- Addition $\Rightarrow +$
- Subtraction $\Rightarrow -$
- Scalar multiplication $\Rightarrow *$
- Division $\Rightarrow /$
- Raise to power $\Rightarrow \wedge$
- Matrix multiplication $\Rightarrow .$

Arithmetic operations

Examples

(i) 2+6;

(ii) 4-9;

(iii) 5*6;

(iv) 2.45/6.23;

(v) 5*6/3;

(vi) 12^6;

(vii) 3+5*4;

(viii) 2-9+6*5;

(ix) 2-(9+6)*5;

(x) 2-9+6*5+8/2;

Use output for further computations

- The operator % refers to the output expression most recently computed by Maxima, whether or not it was displayed.
- It is not necessarily the content of the output cell just above your current input cell.
- In addition the result of the i-th computation is available by %oi.

Use output for further computations

Examples

```
(%i1) 12+3;
(%o1) 15
(%i2) % * 2;
(%o2) 30
(%i3) % - 10;
(%o3) 20
(%i4) %o1 - 10;
(%o4) 5
```

Number types supported by Maxima

Maxima distinguishes between four different types of numbers:

- Integers.
- Rational numbers.
- Floating point numbers.
- Arbitrary precision floating point numbers (bigfloat numbers).

Number types supported by Maxima

Integers

- Maxima can handle large integers.
- Examples for integers are: $-4, 3, 2, 1, 0, 1, 2, 3, 4, \ldots$
- You can use $\mathbf{12}$ or $\mathbf{12.}$ to enter 12 as an integer in Maxima.
- But $\mathbf{12.0}$ does not represent an integer in Maxima.

Number types supported by Maxima

Integers ⇒ Examples

(i) 15!;

(ii) 13^24;

(iii) 12233.*23334545

(iv) 1223567332/2.

(v) 1223567332/2.0

(vi) 83430290.+5345021144.

Number types supported by Maxima

Rational numbers

- A number which can be written as a ratio of two integers is called as a rational number.
- Examples for rational numbers are: $23/3, 5/2, -13/4....$
- You can use **23/3, 23./3, 23/3.** or **23./3.** to enter 23/3 as a rational number in Maxima.
- But **23.0/3, 23/3.0** or **23.0/3.0** do not represent a rational number in Maxima.

Number types supported by Maxima

Floating point numbers

- These numbers consist of a *mantissa* of (approximately) 16 decimal digits and an exponent to base 10.
- Eg: $1.234567890123456 \times 10^5$.
- In common speech these are called decimal numbers and usually written without the exponent.
- That is, $1.234567890123456 \times 10^5 \rightarrow 123456.7890123456$.
- Floating point numbers can be entered either as a decimal number with at least one digit after the decimal point, e.g., 123.0, or using the scientific notation, e.g., 123e0.

Number types supported by Maxima

Floating point numbers ⇒Rational numbers and floating point numbers

The following example demonstrates the difference between rational numbers and floating point numbers.

(%i1) 2/10 * 11 - 2 - 4/20;
(%o1) 0

(%i2) 2.0/10.0 * 11.0 - 2.0 - 4.0/20.0;
(%o2) $1.665334536937735 \times 10^{-16}$

Number types supported by Maxima

Floating point numbers⇒Remark

- Integers and rational numbers are stored without loss of precision while this is not possible for floating point numbers.
- They can be seen as an approximation to real numbers.
- Notice that the decimal expression of real numbers may have an infinite number of digits as in $\sqrt{2} = 1.414213562373095....$

Number types supported by Maxima

Floating point numbers⇒Remark⇒Cont...

- When stored as floating point numbers only a limited number of digits can be stored and one looses precision.
- Additions and subtractions of floating point numbers then may results in further loss of precision due to cancellation errors.
- To overcome this we have to introduce new number type.

Number types supported by Maxima

Arbitrary precision floating point numbers

- It is also called as **bigfloat** numbers.
- Floating point numbers where the size of the *mantissa* can be set to some fixed but arbitrary number.
- The system variable **fpprec** can be used to set fixed arbitrary number for *mantissa*.

Special nature of Maxima's output

- Maxima tries to do all its evaluation as exact as possible.
- Maxima reduces rational numbers or simplifies numerical expression where possible but does not convert to floating point numbers unless forced to do so.
- In particular Maxima also returns special numbers as results of computations.

Special nature of Maxima's output

Examples

(i) 17/4;

(ii) 3^700;

(iii) sqrt(2);

(iv) 18/4;

(v) sqrt(12);

(vi) exp(3);

(vii) sqrt(8);

(viii) atan(1);

(ix) tan(%pi/4);

(x) $1/101 + 1/101$

Special nature of Maxima's output

Remark 1

- When we use floating point numbers instead, we get a less precise result, i.e., stored as floating point numbers.
- It is often sufficient to insert just one floating point number in order to obtain a floating point answer.

Special nature of Maxima's output

Remark 1⇒ Examples

(i) 17.0/4;

(ii) 3.0^700;

(iii) sqrt(2.0);

(iv) 18.0/4;

(v) sqrt(12.0);

(vi) exp(3.0);

(vii) sqrt(8.0);

(viii) atan(1.0);

(ix) tan(%pi/4.0);

(x) $1.0/101 + 1.0/101$

Special nature of Maxima's output

Remark 2

- Sometimes it can be annoying when 16 digits of floating point numbers are printed.
- This can be controlled by setting system variable **fpprintprec**.

```
(%i21) fpprintprec: 4$
(%i22) sqrt(2.0);
(%o22) 1.141

(%i23) fpprintprec: 0$
(%i24) sqrt(2.0);
(%o24) 1.414213562373095
```

Data type conversion

- Instead of getting a rational form result, we can get numeric results using system variable **numer**.
- An alternative approach is to use the **float** command.

Data type conversion

Examples

(i) 19/3, numer;
(ii) sin(4), numer;
(iii) sqrt(2), numer;
(iv) %pi, numer;
(v) exp(3),numer;
(vi) float(19/3);
(vii) float(sin(4));
(viii) float(%pi);

Data type conversion

Cont...

- wxMaxima tries to print Maximas output in a nice manner.
- Where numbers are printed into one line.
- Suppose you need all digits of 100! or the first 500 digits of π.
- Then not all digits are displayed which may not be what you want.

Data type conversion

Cont...

```
(%i8) 100!;
(%o8) 93326215443944152681699238856[98 digits]916864000000000000000000000000

(%i9) fpprec: 500$
(%i10) bfloat(%pi);
(%o10) 3.1415926535897932384626433832[443 digits]8857527248912279381830119491b0

(%i11) reset()$
```

Data type conversion

Cont...

- However, it is possible to switch back to Maxima's native output format using command **set_display(ascii)**.
- Then all digits are printed as the output.
- The backslash sign at the end of each line indicates that it is continued on the next.
- Do not forget to reset the display format again by means of **set_display(xml);**

Data type conversion

Cont...

```
(%i12) set_display(ascii)$
(%i13) 100!;
(%o13) 93326215443944152681699238856266700490715968264381621468592963895217599\
99322991560894146397615651828625369792082722375825118521091686400000000000000000\
000000000
```

```
(%i14) fpprec: 500$
(%i15) bfloat(%pi);
(%o15) 3.1415926535897932384626433832795028841971693993751058209749445923078164\
062862089986280348253421170679821480865132823066470938446095505822317253594081\
284811174502841027019385211055596446229489549303819644288109756659334461284756\
482337867831652712019091456485669234603486104543266482133936072602491412737245\
870066063155881748815209209628292540917153643678925903600113305305488204665213\
841469519415116094330572703657595919530921861173819326117931051185480744623799\
627495673518857527248912279381830119491b0
```

```
(%i16) reset()$
(%i17) set_display(xml)$
```

Standard functions

Constant functions in Maxima

Maxima knows important numerical constants like e and π as well as $\pm\infty$.

Constant	**Description**
%e	Eulers number $e = 2.71828\ldots$
%pi	$\pi = 3.14159\ldots$
%i	Imaginary unit i$=\sqrt{-1}$
inf	Positive infinity ∞
minf	Minus infinity $-\infty$

Commonly used functions in Maxima

Functions	Description
abs(x)	Absolute value of x
sqrt(x)	Square root of x
log(x)	Natural logarithm (i.e, to base e) of x
exp(x)	Exponential function of x
sin(x)	Sine of x
cos(x)	Cosine of x
tan(x)	Tangent of x

Commonly used functions in Maxima

Examples

(i) sin(%pi);
(ii) cos(%pi/4);
(iii) exp(2);
(iv) abs(-9.899);
(v) tan(%pi/3);
(vi) cot(%pi/2);

Commonly used functions in Maxima

Arguments for trigonometric functions

- The angles as arguments for trigonometric functions must be given in radians.
- To do computations using degrees, first you have to convert degree D into radian R using,

$$R = D\frac{\pi}{180}.$$

- $\sin 30^\circ \Rightarrow \sin(\%\mathrm{pi}/6);$.
- $\cos 45^\circ \Rightarrow \cos(\%\mathrm{pi}/4);$.

Commonly used functions in Maxima

Common and natural logarithms

- The logarithm with base e is called as **natural logarithm**.
- Any positive number is suitable as the base of logarithms.
- Base 10 is used more than any others.
- The logrithm with base 10 is called as **common logarithm**.

Commonly used functions in Maxima

Common and natural logarithms⇒Cont...

- Maxima only provides the natural logarithm function.
- The common logarithm can be computed using,

$$\log_{10}(x) = \frac{\log x}{\log 10}.$$

- Don't use $\ln(x)$ to compute the natural logarithm.
- Maxima does not know $\ln(x)$ function and thus it returns it unevaluated.

Commonly used functions in Maxima

Common and natural logarithms⇒Examples

(i) log(%e);

(ii) Try ln(%e);

(iii) Calculate $\log_{10}(5)$;

(iv) Calculate $\log_{2.1}(3)$;

(v) Calculate $\log_{1.2344}(4)$;

Functions for Numbers

abs (*expr*)

- Returns the absolute value *expr*.
- If *expr* is complex, returns the complex modulus of *expr*.
- Try followings with the function **abs(expr)**.

(i) 20.34;
(ii) -299.34;
(iii) 5i+4;
(iv) -2i-9;

Functions for Numbers

ceiling *(x)*

- When x is a real number, return the least integer that is greater than or equal to x.
- If x is a constant expression ceiling evaluates x using big floating point numbers, and applies **ceiling** to the resulting big float.
- Try followings with the function **ceiling(x)**.

(i) 9.00001;
(ii) 9.99999;
(iii) -9.00001;
(iv) -9.99999;
(v) 14 * %pi;
(vi) -14 * %pi;

Functions for Numbers

entier*(x)*

- Returns the largest integer less than or equal to x where x is numeric.
- **fix(x)** is a synonym for **entier (x)**.
- Try followings with the function **entier(x)**.

  (i) 9.00001;
  (ii) 9.99999;
  (iii) -9.00001;
  (iv) -9.99999;

Functions for Numbers

floor *(x)*

- When x is a real number, return the largest integer that is less than or equal to x.
- If x is a constant expression, **floor** evaluates x using big floating point numbers, and applies **floor** to the resulting big float.
- Try followings with the function **floor(x)**.

(i) 9.00001;
(ii) 9.99999;
(iii) -9.00001;
(iv) -9.99999;
(v) 14 * %pi;

Random number generation

- The function **random** (x) is used for random number generation.
- If x is an integer, **random** (x) returns an integer from 0 through $x - 1$ inclusive.
- If x is a floating point number, **random** (x) returns a nonnegative floating point number less than x.
- It complains with an error if x is neither an integer nor a float, or if x is not positive.

The use of variables and user defined functions

Variables and variables names in Maxima

- A variable is a symbolic name associated with a value and whose associated value may be changed.
- The alphanumeric characters are A through Z, a through z, 0 through 9,
- A valid variable name should be started with a letter and any alphanumeric characters can be used as remainings.
- Valid variable names are: c, X, *age_of_male*, or y_1.
- Maxima is case-sensitive, that is, the identifiers **to**, **TO**, and **To** are distinct.

Assignment statements

- An assignment statement sets or re-sets the value stored in the storage location(s) denoted by a variable name.
- : operator is used in Maxima for assignment.
- This operator evaluates its right-hand side and associates that value with the left-hand side.
- When the variable is evaluated in further computations, then it is replaced by its value.
- It is not possible to use $=$ operator for assigning value to a variable.

Assignment statements

Example

(%i1) y;
(%o1) y
(%i2) $y : 20$;
(%o2) 20
(%i3) y;
(%o3) 20
(%i4) $L : 2 * y^2$;
(%o4) 800
(%i5) $L + 5$;
(%o5) 805
(%i6) $L : y$;
(%o6) 20

Substitution

- The command **subst(**$a = b, expr$**);** substitutes the expression b for the variable a in the expression $expr$.
- To perform multiple substitutions use **subst(**$[eqn\_1, .., eqn\_n], expr$**);** where each of the eqn_i are equations indicating the substitutions to be made.

Substitution

Examples

1. Let $f : \sin((x + y + z)/2);$. Subsitute the value of $z = 10$.
2. In the above function subsitute the value of $x = \cos(a + b)$.
3. Let $c : a + b;$. Subsitute $a = 10$ and $b = 12$.

User defined functions

- By using function definition operator :=, it is possible to define our own functions in Maxima.
- Function names are similar to variable names but are followed by parenthesis (...) that contain a comma separated list of its arguments.
- The right hand side of the function assignment operator := (i.e., the function body) is never evaluated.

User defined functions
Examples

Define functions for followings.

(i) To compute the square of a given expression.

(ii) To calculte cos value when the angle is given in degrees.

Clear user defined variables and functions

- The system variables **values;** and **functions;** contain a list of user defined variables and functions, respectively.
- Both variables and functions remain persistent until the Maxima session is closed.
- Sometimes it is convenient to remove some unuseful variables and functions.
- It can be accomplished by using function **kill**.

System variables

- Maxima uses a set of system variables to control the behavior of the system.
- For example, as mentioned above variable **fpprintprec** is used to control the printing of floating point numbers.
- And also **numer** controls whether mathematical functions are evaluated in floating point or not.

System variables

Reset system variables

- One may use assign operator **:** to change value of system variables globally.
- Command **reset()** allows to reset many global system variables and some other variables, to their default values.

System variables

Reset system variables locally

- An alternative approach to changing and resetting system variables is the use of command **ev**.
- Which allows to evaluate an expression with locally changed system variables.
- Try **ev(17/3, numer:true);**.

Thank you!