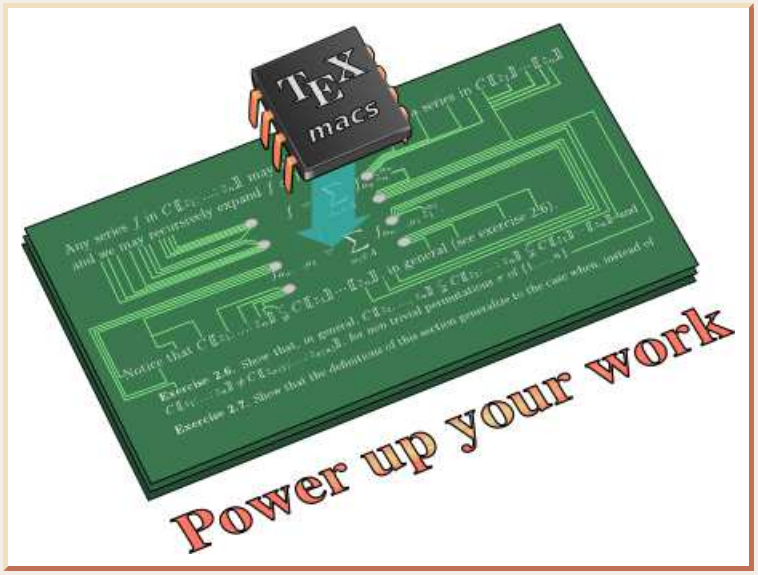
# Introduction to Mathemagix

Joris van der Hoeven, Grégoire Lecerf, Bernard Mourrain, ...




Auron 2010

http://www.TEXMACS.org

http://www. .org

# Overview

- **Aims**
  - Strongly typed language with a compiler.
  - Access to low level details and powerful abstractions.
  - Well integrated with other languages, such as C++.
  - Libraries for computer algebra and computer analysis.
  - User friendly interface based on GNU TEX$_{\text{MACS}}$.

- **Current status**
  - Interpreter with inperfect typing.
  - Experimental compiler under development.
  - Efficient C++ libraries for computer algebra and analysis.

# Running the interpreter

```
Mmx] 1 + 1
```

```
2
```

```
Mmx] "Hello" >< "There"
```

```
"HelloThere"
```

```
Mmx] for i in 1 to 5 do mmout << i << " -> " << i*i << "\n";
```

```
1 -> 1
2 -> 4
3 -> 9
4 -> 16
5 -> 25
```

```
Mmx] type_mode? := true;
Mmx] 1 + 1
```

2: Int

```
Mmx] use "algebramix";
Mmx] 1 + 1
```

2: Integer

```
Mmx] z == series (0, 1)
```

$z + O(z^{10})$: Series(Rational)

```
Mmx] sin z
```

$z - \frac{1}{6}z^3 + \frac{1}{120}z^5 - \frac{1}{5040}z^7 + \frac{1}{362880}z^9 + O(z^{10})$: Series(Rational)

```
Mmx] help series
```

```
series : Tuple (Generic) -> Series (Generic)                                  (Native)
series : Tuple (Modular (Integer)) -> Series (Modular (Integer))              (Native)
series : Tuple (Rational) -> Series (Rational)                                (Native)
series : Tuple (Complex (Rational)) -> Series (Complex (Rational))            (Native)
series : Tuple (Unknown (Rational)) -> Series (Unknown (Rational))  (Native): Document
```

```
Mmx] help Series Rational
```

Series (Rational) (Class) : Document

**Available functions**

```
!= : (Series (Rational), Series (Rational)) -> Boolean
* : (Series (Rational), Series (Rational)) -> Series (Rational)
* : (Rational, Series (Rational)) -> Series (Rational)
* : (Series (Rational), Rational) -> Series (Rational)
+ : (Series (Rational), Series (Rational)) -> Series (Rational)
+ : (Rational, Series (Rational)) -> Series (Rational)
+ : (Series (Rational), Rational) -> Series (Rational)
- : Series (Rational) -> Series (Rational)
- : (Series (Rational), Series (Rational)) -> Series (Rational)
- : (Rational, Series (Rational)) -> Series (Rational)
- : (Series (Rational), Rational) -> Series (Rational)
.[] : (Series (Rational), Int) -> Rational
.[] : (Series (Rational), Int, Int) -> Polynomial (Rational)
/ : (Series (Rational), Series (Rational)) -> Series (Rational)
/ : (Rational, Series (Rational)) -> Series (Rational)
/ : (Series (Rational), Rational) -> Series (Rational)
<< : (Series (Rational), Int) -> Series (Rational)
<= : (Series (Rational), Series (Rational)) -> Boolean
< : (Series (Rational), Series (Rational)) -> Boolean
= : (Series (Rational), Series (Rational)) -> Boolean
>= : (Series (Rational), Series (Rational)) -> Boolean
>> : (Series (Rational), Int) -> Series (Rational)
> : (Series (Rational), Series (Rational)) -> Boolean
@ : (Series (Rational), Series (Rational)) -> Series (Rational)
^ : (Series (Rational), Int) -> Series (Rational)
^ : (Series (Rational), Integer) -> Series (Rational)
^ : (Series (Rational), Series (Rational)) -> Series (Rational)
```

```
Mmx]
```

# Running the compiler

## ⇑ Hello world

```
Shell] mmc
```

```
Usage: mmc [options] input_file [-o output_file]

Supported options:
  --compile    Compile only and don't link
  --debug      Print debugging information
  --gdb        Add debugging information for gdb
  --keep       Keep the temporarily generated C++ file
  --no-cache   Do not use cached information for fast compilation
  --optimize   Compile with C++ optimization flags
  --threads n  Compile using at most n threads
  --static     Create a static binary
  --verbose    Print information about compilation stages
```

```
Shell] cat hello.mmx
```

```
mmout << "Hello world\n";
```

```
Shell] mmc hello.mmx
```

```
Shell] ./hello
```

```
Hello world
```

```
Shell]
```

## ⇑ Separate compilation

```
Shell] rm -rf ~/.mathemagix/mmc
Shell] cat slave.mmx
Shell] cat master.mmx
Shell] mmc --verbose master.mmx
mmc: analyzing dependencies
mmc: compiling /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/utilities/
builtin.mmx
mmc: compiling /Users/vdhoeven/seminar/Auron/slave.mmx
mmc: c++ `mmcompiler-config --cppflags` -c /Users/vdhoeven/mathemagix/mmx/
mmcompiler/compiler/utilities/builtin.cpp -o /Users/vdhoeven/mathemagix/mmx/
mmcompiler/compiler/utilities/builtin.o
mmc: c++ `mmcompiler-config --cppflags` -c slave.cpp -o slave.o
mmc: compiling /Users/vdhoeven/seminar/Auron/master.mmx
mmc: c++ `mmcompiler-config --cppflags` -c master.cpp -o master.o
mmc: attaching /Users/vdhoeven/seminar/Auron/master-main.cpp
mmc: c++ -c master-main.cpp -o master-main.o
mmc: c++ -o master /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/
utilities/builtin.o slave.o master.o master-main.o `mmcompiler-config --
libs`
Shell] ./master
1 -> 2
2 -> 9
3 -> 28
4 -> 65
5 -> 126
6 -> 217
```

# Atomic types

## ⇑ Booleans

```
Mmx] type_mode? := true;
Mmx] 2 = 2
true: Boolean
Mmx]
```

## ⇑ Strings

```
Mmx] "ho" >< "ho"
```

"hoho": String

```
Mmx] replace ("hoho", "ho", "haha")
```

"hahahaha": String

```
Mmx]
```

## ⇑ Syntactic types

```
Mmx] 'x
```

$x$: Literal

```
Mmx] '(f (x, y, z))
```

$f(x, y, z)$: Compound

```
Mmx] '(f (x, y, z)) [2]
```

$y$: Literal

```
Mmx] $document ("Some ", $with ("color", "red", "red"), " text.";
                "Pythagoras said ", $math ('(a^2 + b^2 = c^2)), ".")
```

Some red text.
Pythagoras said $a^2 + b^2 = c^2$.: Document

```
Mmx] mmout << $document ("More ", $strong ("very important"), " stuff.") << "\n";
```

More **very important** stuff.

```
Mmx]
```

## ⇑ Ports

```
Mmx] mmout
```

formatting_port(output_file_port(mmout)): Port

```
Mmx] mmout << "Hello\n";
```

```
Hello
```

```
Mmx] output_file_port ("toto.txt") << "Hi there\n";
Mmx] load ("toto.txt")
```

"Hi there\n": String

```
Mmx]
```

# Numeric types

## ⇑ Integers

```
Mmx] use "numerix";
Mmx] 100!
```

```
933262154439441526816992388562667004907159682643816214685929638952175999932299156089\
4146397615651828625369792082722375825118521091686400000000000000000000000: Integer
Mmx]
```

## ⇑ Rational numbers

```
Mmx] 1/1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + 1/9 + 1/10
```

$\frac{7381}{2520}$: Rational

```
Mmx]
```

## ⇑ Floating point numbers

```
Mmx] exp 1.0
```

2.718281828459045235: Floating

```
Mmx] bit_precision := 128;
Mmx] exp 1.0
```

2.71828182845904523536028747135266249759: Floating

```
Mmx] significant_digits := 0
```

0: Int

```
Mmx] exp 1.0
```

2.7183: Floating

```
Mmx] exp 1.0000000000000000000000001 - exp 1.0
```

$2.7183e-25$: Floating

```
Mmx]
```

## ⇑ Complex numbers

```
Mmx] complex (1, 2/3)
```

$1 + \frac{2}{3}\mathrm{i}$: Complex(Rational)

```
Mmx] square complex (1, 2/3)
```

$\frac{5}{9} + \frac{4}{3}\mathrm{i}$: Complex(Rational)

```
Mmx]
```

## Tangent numbers

```
Mmx] tangent (1, 2)
```

tangent$(1, 2)$: Tangent(Rational)

```
Mmx] tangent (1, 2) * tangent (1, 2)
```

tangent$(1, 4)$: Tangent(Rational)

```
Mmx]
```

## Balls

```
Mmx] b == ball (1.0, 0.00000000001)
```

$1.0000000000$: Ball(Floating)

```
Mmx] 1.0001 * b - b
```

$1.000000e-4$: Ball(Floating)

```
Mmx]
```

# Compound types and syntactic sugar

## ⇑ Vectors

```
Mmx] use "algebramix";
Mmx] v == [1, 2, 3, 4, 5, 6]
```

$[1, 2, 3, 4, 5, 6]$: Vector(Integer)

```
Mmx] v >< map (square, v)
```

$[1, 2, 3, 4, 5, 6, 1, 4, 9, 16, 25, 36]$: Vector(Integer)

```
Mmx] v*v + 3*v + 2
```

$[6, 12, 20, 30, 42, 56]$: Vector(Integer)

```
Mmx] #v
```

6: Int

```
Mmx] v[1]
```

2: Integer

```
Mmx] v[2,4]
```

$[3, 4]$: Vector(Integer)

```
Mmx]
```

## ⇑ Iterators and syntactic sugar

```
Mmx] 1 to 10
```

iterator(1, 2, 3, 4, 5, 6, 7, 8, 9, 10): Generator(Generic)

```
Mmx] [ 1 to 10 ]
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]: Vector(Integer)

```
Mmx] [ 0..10 ]
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]: Vector(Integer)

```
Mmx] [ k^3 | k in 1 to 10 ]
```

[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]: Vector(Integer)

```
Mmx] [ k^3 | k in 1 to 10, k mod 3 = 1 ]
```

[1, 64, 343, 1000]: Vector(Integer)

```
Mmx]
```

## ⇑ Matrices

```
Mmx] [ 1, 2; 3, 4 ]
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} : \text{Matrix(Integer)}$$

```
Mmx] M == [ 1 / (i+j+1) | j in 0..4 || i in 0..4 ]
```

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix} : \text{Matrix(Rational)}$$

```
Mmx] invert M
```

$$\begin{bmatrix} 16 & -120 & 240 & -140 \\ -120 & 1200 & -2700 & 1680 \\ 240 & -2700 & 6480 & -4200 \\ -140 & 1680 & -4200 & 2800 \end{bmatrix} : \text{Matrix(Rational)}$$

```
Mmx]
```

## ⇑ Tables

```
Mmx] t := table (0)
```

table(): Alias(Generic)

```
Mmx] t['x]
```

0: Alias(Generic)

```
Mmx] t['x] := "Hallo";
Mmx] t
```

table([$x$, "Hallo"]): Alias(Generic)

```
Mmx] t['x]
```

"Hallo": Alias(Generic)

```
Mmx] t[polynomial(1,1)^10] := "Boeh";
Mmx] t
```

table([$x$, "Hallo"], [$x^{10} + 10\,x^9 + 45\,x^8 + 120\,x^7 + 210\,x^6 + 252\,x^5 + 210\,x^4 + 120\,x^3 + 45\,x^2 + 10\,x + 1$, "Boeh"]): Alias(Generic)

```
Mmx]
```

# Simple declarations

## ⇑ Constant and mutable variables

```
Mmx] cst: Integer == 111111111^2
```

12345678987654321: Integer

```
Mmx] mut: Integer := 10101010101010101010101010101^2
```

1020304050607080910111213141516151413121110090807060504030201: Alias(Integer)

```
Mmx] cst := square cst
```

```
1:1: exception, expected Alias \/ Tuple
cst := square cst
~~~
```

```
Mmx] mut := square mut
```

1083723380950975137713575001030581070132377965655195420391149074422829950142281896938\
352610493465982566110520916971785131887567059999977756499679767804637610614602670586 6\
250841235445081186719343318240678970866146027081963046949239531203608 01: Integer

```
Mmx]
```

## ⇑ Function definitions and overloading

```
Mmx] cube (x: Integer): Integer == x*x*x;
Mmx] cube (s: String): String == s >< s >< s;
Mmx] cube 10010010010010001
100300601001502102502702702502101501000600300l: Integer
Mmx] cube "haha"
"hahahahahaha": String
Mmx]
```

## ⇑ The generic type

Disclaimer: special arithmetic on generic objects only available in interpreter.

```
Mmx] haha (ha) == ha * ha;
```

```
Mmx] haha 1111
```

1234321: Integer

```
Mmx] infix * (s1: String, s2: String): String == s1 >< s2;
```

```
Mmx] haha "ahah"
```

"ahahahah": String

```
Mmx] haha (ha: Integer) == ha * ha * ha * ha;
```

```
Mmx] haha 2
```

16: Integer

```
Mmx] haha "ohoh"
```

"ohohohoh": String

```
Mmx]
```

```
Mmx] fib (n: Integer): Integer ==
       if n <= 1 then 1 else fib (n-1) + fib (n-2);
Mmx] [ fib n | n in 1 to 20 ]
```

[1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946]: Vector(Integer)

```
Mmx] shift (x: Integer) (y: Integer): Integer == x + y;
Mmx] shift 3
```

closure: Function

```
Mmx] (shift 3) 4
```

7: Integer

```
Mmx] map (shift 3, [ 1 to 20 ])
```

[4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]: Vector(Integer)

```
Mmx]
```

# Class definitions

```
Mmx] use "numerix"; type_mode? := true; significant_digits := 5;
```

```
Mmx] class Color == {
       mutable { r: Floating; g: Floating; b: Floating; }
       constructor grey (x: Floating) == {
         r == x; g == x; b == x; }
       constructor rgb (r2: Floating, g2: Floating, b2: Floating) == {
         r == r2; g == g2; b == b2; }
     }
```

```
Mmx] rgb (1, 0, 0)
```

object([1.0000, 0, 0], Color): Color

```
Mmx] flatten (c: Color): Syntactic ==
       syntactic ('rgb (as_generic flatten c.r,
                        as_generic flatten c.g,
                        as_generic flatten c.b));
```

```
Mmx] rgb (1, 0, 0)
```

rgb(1.0000, 0, 0): Color

```
Mmx] mix (c1: Color, c2: Color, a: Floating): Color ==
       rgb (a * c1.r + (1-a) * c2.r,
            a * c1.g + (1-a) * c2.g,
            a * c1.b + (1-a) * c2.b);
```

```
Mmx] mix (rgb (1, 0, 0), grey (0.5), 0.5)
```

rgb(0.75000, 0.25000, 0.25000): Color

```
Mmx]
```

# Implicit type conversion and inheritance

```
Mmx] upgrade (x: Floating): Color == grey x;
```

```
Mmx] mix (1.0, rgb (0, 1, 0), 0.2)
```

rgb(0.20000, 1.0000, 0.20000): Color

```
Mmx] mix (1, rgb (0, 1, 0), 0.4)
```

rgb(0.40000, 1.0000, 0.40000): Color

```
Mmx] class Alpha_color == {
       mutable { c: Color; a: Floating; }
       constructor alpha_color (c2: Color, a2: Floating) == { c == c2; a == a2; }
     };
```

```
Mmx] alpha_color (0, 0.5)
```

object([rgb(0, 0, 0), 0.50000], Alpha_color): Alpha_color

```
Mmx] downgrade (ac: Alpha_color): Color == mix (ac.c, 1, ac.a);
```

```
Mmx] alpha_color (0, 0.5) :> Color
```

rgb(0.50000, 0.50000, 0.50000): Color

```
Mmx] postfix .greyed (c: Color): Color == (c.r + c.g + c.b) / 3;
```

```
Mmx] rgb (1, 0, 0).greyed
```

rgb(0.33333, 0.33333, 0.33333): Color

```
Mmx]
```

# Control structures



## ⇑ Loops

```
Mmx] for i in [ 1, 4, 7] for j in [ 3, 5, 11 ] do
       mmout << i << ", " << j << " -> " << i * j << "\n";
```

```
1, 3 -> 3
4, 5 -> 20
7, 11 -> 77
```

```
Mmx] for i: Int in 0..100 until i*i >= 5000 do {
       if i mod 11 <= 8 then continue;
       mmout << "i= " << i << "\n";
     }
```

```
i= 9
i= 10
i= 20
i= 21
i= 31
i= 32
i= 42
i= 43
i= 53
i= 54
i= 64
i= 65
```

```
Mmx]
```

## ⇑ Exceptions

```
Mmx] risky (x: Rational): Rational == {
       if x = 5 then raise "not in domain";
       return 1 / (x - 5);
     }
     try {
       for i in 1 to 10 do mmout << i << " -> " << risky i << "\n";
       catch (err: String) { mmout << "error: " << err << "\n"; }
     };
```

$1$ -> $\frac{-1}{4}$
$2$ -> $\frac{-1}{3}$
$3$ -> $\frac{-1}{2}$
$4$ -> $-1$
$5$ -> error: not in domain

```
Mmx]
```

# Categories

```
Shell] mmc --verbose --keep ring.mmx
```

```
mmc: analyzing dependencies
mmc: compiling /Users/vdhoeven/seminar/Auron/ring.mmx
mmc: compiling /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/utilities/
builtin.mmx
mmc: c++ `mmcompiler-config --cppflags` -c ring.cpp -o ring.o
mmc: attaching /Users/vdhoeven/seminar/Auron/ring-main.cpp
mmc: c++ -c ring-main.cpp -o ring-main.o
mmc: c++ -o ring /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/utilities/
builtin.o ring.o ring-main.o `mmcompiler-config --libs`
```

```
Shell] ./ring
```

```
1 -> 101
2 -> 104
3 -> 109
4 -> 116
5 -> 125
6 -> 136
7 -> 149
8 -> 164
9 -> 181
10 -> 200
```

```
Shell]
```

```
Shell] mmc --verbose --keep recursive.mmx
mmc: analyzing dependencies
mmc: compiling /Users/vdhoeven/seminar/Auron/recursive.mmx
mmc: c++ `mmcompiler-config --cppflags` -c recursive.cpp -o recursive.o
mmc: attaching /Users/vdhoeven/seminar/Auron/recursive-main.cpp
mmc: c++ -c recursive-main.cpp -o recursive-main.o
mmc: c++ -o recursive /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/
utilities/builtin.o recursive.o recursive-main.o `mmcompiler-config --libs`
Shell] ./recursive
[1, 2, 3, 8, 15, 30, 56, 104, 189, 340]
Shell]
```

# Gluing C++ functionality



```
Shell] mmc --verbose syracuse.mmx
```

```
mmc: analyzing dependencies
mmc: compiling /Users/vdhoeven/mathemagix/mmx/numerix/mmx/integer.mmx
mmc: compiling /Users/vdhoeven/seminar/Auron/syracuse.mmx
mmc: c++ `mmcompiler-config --cppflags` `numerix-config --cppflags` -c /Users/
vdhoeven/mathemagix/mmx/numerix/mmx/integer.cpp -o /Users/vdhoeven/mathemagix/
mmx/numerix/mmx/integer.o
mmc: c++ `mmcompiler-config --cppflags` `numerix-config --cppflags` -c
syracuse.cpp -o syracuse.o
mmc: attaching /Users/vdhoeven/seminar/Auron/syracuse-main.cpp
mmc: c++ -c syracuse-main.cpp -o syracuse-main.o
mmc: c++ -o syracuse /Users/vdhoeven/mathemagix/mmx/mmcompiler/compiler/
utilities/builtin.o /Users/vdhoeven/mathemagix/mmx/numerix/mmx/integer.o
syracuse.o syracuse-main.o `numerix-config --libs` `mmcompiler-config --libs`
```

```
Shell] ./syracuse
```

[11111111111111111111111111111111, 33333333333333333333333333333334,
16666666666666666666666666666667, 50000000000000000000000000000002,
25000000000000000000000000000001, 75000000000000000000000000000004,
37500000000000000000000000000002, 18750000000000000000000000000001,
56250000000000000000000000000004, 28125000000000000000000000000002,
14062500000000000000000000000001, 42187500000000000000000000000004,
21093750000000000000000000000002, 10546875000000000000000000000001,
31640625000000000000000000000004, 15820312500000000000000000000002,
79101562500000000000000000000001, 23730468750000000000000000000004,
11865234375000000000000000000002, 59326171875000000000000000000001,
17797851562500000000000000000004, 88989257812500000000000000000002,
44494628906250000000000000000001, 13348388671875000000000000000004,
66741943359375000000000000000002, 33370971679687500000000000000001,
10011291503906250000000000000004, 50056457519531250000000000000002,
25028228759765625000000000000001, 75084686279296875000000000000004,
37542343139648437500000000000002, 18771171569824218750000000000001,
56313514709472656250000000000004, 28156757354736328125000000000002,
14078378677368164062500000000001, 42235136032104492187500000000004,
21117568016052246093750000000002, 10558784008026123046875000000001,
31676352024078369140625000000004, 15838176012039184570312500000002,
79190880060195922851562500000001, 23757264018058776855468750000004,
11878632009029388427734375000002, 59393160045146942138671875000001,
17817948013544082641601562500004, 89089740067720413208007812500002,
44544870033860206604003906250001, 13363461010158061981201171875004,
66817305050790309906005859375000, 20045191515237092971801757813200,
10022595757618546485900878906600, 50112978788092732429504394533000,
15033893636427819728851318360000, 75169468182139098644256591800000,
37584734091069549322128295900000, 18792367045534774661064147950000,
93961835227673873305320739750000, 28188550568302161991596221926000,

```
Shell]
```