New Scientific Programming Language

...

Taha BEN SALAH (NOCCS, ENISo, Sousse University)
Taoufik AGUILI (Sys'Com, ENIT, UTM University)

Agenda

- Introduction
- Existing Scientific Programming Languages
- Presenting the new Scientific Programming Language
- Comparisons and results
- Conclusion

What is a scientific language

A scientific language is a programming language optimized for the use of mathematical formulas and matrices.

Existing Scientific Languages:
DSL : Domain Specific Languages

MATLAB, Maple, FORTRAN, ALGOL, APL,J, Julia, Wolfram Language/Mathematica, and R.

Non Scientific Languages used by scientists
GPL: General Purpose Languages

C/C++, Python, Scala, Java

Why a new Programming Language

- Limits of the existing Programming Languages
- Simpler syntax or More Powerful syntax
- Better portability
- Better integration
- Better performance

Existing Languages

	Python	Java/Scala/Kotlin	Julia	Matlab	R
Type	GPL	GPL	DSL	DSL	DSL
Paradigm	Prototyping, Web, Data Science	Server, Big Data, Distributed Systems	Computation	Scientific computation, Matrices, ...	Statistics, Data Science
Price	Free & Open Source	Free & Open Source	Free & Open Source	Commercial	Free & Open Source
Advantages	Community, Simplicity, Libraries	Community, Tools, Libraries	Simplicity, Performance	Simplicity, Toolboxes (Simulink)	Toolboxes
Limitations	Performance, tools, compatibility	Steep learning curve	Small popularity, few libraries	Commercial, not appropriate for big projects, performance	not appropriate for complex projects

Existing Languages

	Python	Java/Scala/Kotlin	Julia	Matlab	R
Type	Dyn Typed	Static Typed	Dyn Typed	Dyn Typed	Dyn Typed
Paradigm	OO / Proc	OO / Func	Proc, loosely OO/Func	Proc - Loosely OO	Proc - Loosely OO
Compiled?	Interpreted + compiled	Compiled+JIT	compiled+JIT	Interpreted + compiled	Interpreted
Toolchain	REPL, Interpreter + IDE	Compiler+Externa Build Tools	Compiler	Studio	Studio
Dependencies & Versioning	Strong support	Strong support	Weak Support	Weak Support	Weak Support

Existing Languages

	Python	Java/Scala/Kotlin	Julia	Matlab	R
Parallel / Concurrent / Distributed	Supported	Strongly supported	Supported	Supported	Supported
support CPU/GPU	Via Libraries	JOCL, JogAmp, and JavaCL	Via Libraries	Via Libraries	Via Libraries
Libraries and Ecosystem	Large community	Extremely Large community	Small community	Large community	Large community
Tools support	Good support by Commercial IDE	Best IDEs	Little support	Product Studio	Product Studio
Performance	Bad performance	Good Performance	Better performance	Bad performance	Bad performance

Existing Languages

	Python	Java/Scala/Kotlin	Julia	Matlab	R
Portability	Bad Portability	Better Portability	Supported Win/Linux	Supported Win/Linux	Supported Win/Linux
Numeric Scientific Calculation	Libraries, bad integration	Libraries, bad integration	Complex / Matrices	Complex / Matrices	Complex / Matrices
Symbolic Scientific Calculation	Libraries, bad integration	Libraries, bad integration	Supported	Supported	Supported
Scientific Readability	bad	worst	good	good	average
Native Integration	average	average	good	average	minimal

Journey to a new Programming Language

- Used **Matlab/Scilab/Octave** in simulations
  - As soon as the number of files becomes important (100) this is no more manageable
  - Serious performance isssues
- Moved to **C/C++** (blas, ...)
  - Very hard to maintain too
- Moved to **Java**
  - Rewritten all of the code in Java
  - Written microwaves library
  - Good performance
  - Not accepted complexity by other research colleagues
- Moved to **Scala**
  - Refactored the code into
    - Hadruplots
    - Hadrumaths
    - Hadruwaves
  - Written ports to the libraries in Scala
  - Much better readability
  - Still
    - quite difficult to start a new “code” from scratch
    - Scala is difficult for non initiated to programming researchers
    - Compiling errors difficult to “understand”
    - Limitations inherited from the Java Language
    - Inconsistencies between Scala Collections & Java Collections

Hadra Language

- New programming Language that obviously learns from predecessors
- Focuses on Complex numbers, Vectors, and Matrices
- Base on the Java VM (compiles to Java Byte Code)
- Statically Typed
- Makes advantage of Unicode support

- Concise
- Readable
- Single file project
- Modular & Extensible
- Functional and OO
  - All constructs are functions (for, while, switch...)
  - All functions are Objects

- Introduces ***Elastic Calculation Concept*** : Numeric and Symbolic at the same time
- GPL as DSL

Hadra : Hello World

file: hello.hl

```
println(matrix(3,(i,j)->i+j));
```

shell:

```
> hl hello.hl
```

result:

```
[
  0 1 2
  1 2 3
  3 4 5
]
```

file: plot.hl

```
import net.thevpc.scholar:hadrumaths;
Plot.title("sinus function").asCurve
      .plot(sin(X)*II(0..2π));
```

shell:

```
> hl plot.hl
```

Hadra : Literals

```
int  twelveDecimal = 12 ;
short sixteenBinary = 0b10000s;
bigint  twelveHexBigInt = 0xCI ;
bigdecimal  tens = 10.2E23D;
long  C = 300_000_000 GHz;
var  μ0 =4π*10-7 H/m;
localdate d= t"2020 –02 –01";
Complex c= î+1;
var c2= î+1;
var msg=$"the day is $d";
var json={a:1, b:'two'};
```

Arrays

```
int[5] tab ( i –> 2*i ) ; // [0, 2, 4, 6, 8]
tab[0..2] = [15, 20, 30];   // [15, 20, 30, 6, 8]
tab[2..4] = tab[4..2];    // [15, 20, 8, 6 30]
int[5] tab2(1) ; // [1, 1, 1, 1, 1]
int[5] tab3(Math::random) ; // [0.1, 0.5, 0.2, 0.7,
0.1]
int[5] tab4=[1, 2, 3] ; // [1, 2, 3]
int[] tab5 = tab1 :+ tab2 :+ tab3; // concat
```

Hadra : Functions & Classes

```
fun int  sqsum(int ...a) { // sqsum(1,2)=5
    switch(a.length)
    case 0: 0;  case 1: a[0];
    default : a[0]²+sqsum(a[1..]);}
fun boolean palindrome(int[] a) {
    a[..$/2]==a[$..$/2];}
```

Extension function

```
fun double Complex::norm(this a) {a.abs();}
var c= î+1;
var v1=norm(c); var v2=c.norm();
```

```
class Complex(double r, double i){
    fun double abs(){sqrt(r²+i²);}
    fun Complex  +(Complex o) {
            Complex(r+o.r,i+o.i);          }
    fun Complex  +(double o)
                        {Complex(r+o,i);}
    fun Complex (double o)+
                        {Complex(r+o,i);}
}
```

Hadra : Matrices

```
Matrix<int>  m1 =
                    [0 , 0 , 0 ; 0, 1, 2  ; 0, 2, 4 ];
Matrix<int>  m2 = [ 1 , 2 , 3 ;
                          3 , 2 , 1 ];
var  m3 =matrix(3,(i,j)->i*j);  // = m1
var  m3 =matrix(3,(i,j)->i*î+j);  // complex
matrix
var  m3 =symMatrix(3,(i,j)->i*î+j);
var  v1 =vector(3,(i)->i*î); // vector of
complexes
```

Sums

```
double v=sum(1..1000, x -> sin(x²)) ;
```

Scalar Products

```
int[] tab1 = [a, b, c];
int[] tab2 = [A, B, C];
int v = tab1 ** tab2 ; // = aA+bB+cC
int[][] v2 = tab1 :** tab2 ; // = [a**A , a**B , a**C
                                   b**A , b**B , b**C
                                   c**A , c**B , c**C
                                  ]
```

Hadra : Symbolic Programming

```
// function declared on [0..π], zero elsewhere
var  f = sin(X) *  cos(Y) * II(0..π) ;
// symbolic derive
var  g = derive(f , X) ;
// symbolic integration
var  g = integrate(f , X, 0..π/2) ;
```

```
Param m();
Param n();
var  f□□ = sin(m*X) *  cos(n*Y) * II(0..π)

var  f01=f□□(n->0, m->1)
var all_f=seq(f□□ , n->0..3, m->0..3)
```

Using Latex

```
// ## This is a markdown comment Title
// Here is an example of using latex expressions
var θ=X;
var f1=```latex \cos^2 \theta - \sin^2 \theta```;
var f2=cos(θ)²sin(θ)²;
Plot.plot(f1,f2);
```

Hadra : Elastic Calculation

- The runtime is responsible of switching from Symbolic to Numeric (and vice versa)
- No need to explicitly sym/dblquad (as in matlab)
- Rule Based decisions
- Includes simplications/transformations to detect usual expressions
- Numeric calculation is done only when needed

```
var formal_scalar_product = sin(m*X) **  cos(n*Y) * ∏(0..π);
var formal_scalar_product = sin(X) **  cos(Y) * ∏(0..π);
var numeric_scalar_product = sin((1+X)/cos(Y)) **  cos(Y) * ∏(0..π);
var formal_simplifiable = sin(X)² + cos(X)²;
```

Hadra Integration : Native Code (C/C++), Java / Scala

- Seamless integration with C/C++
- Uses JNA/JNA under the hoods
- Uses a specific annotation @native

```
@native("c")
class StandardAccess {
    void printf(string format, object ... args);
    int scanf(string format, object ... args);
}
byte[32] bytes;
StandardAccess.scanf("%s", bytes);
StandardAccess.printf("%s [%s] %s %s!\n", "Your message", Native.toString(bytes), "is printed in C", "5.5.0");
```

- Seamless integration with Java/Scala

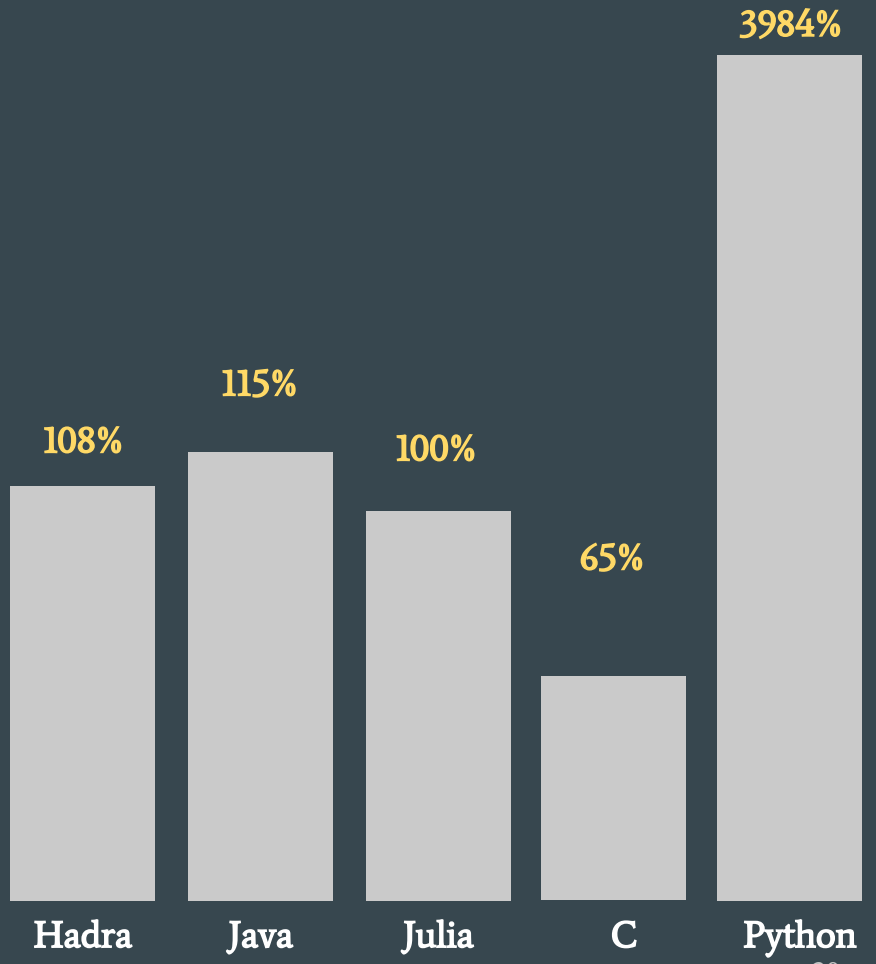
```
JFrame f("java frame")
        .{visible=true; title="example";};
f+Button("click me");
```

Hadra vs the world

	Hadra	Java	Julia	C/C++	Python	Matlab
Operator overloading	yes**	no	yes	yes	yes	no
Superscript/Subscript	yes	no	no	no	no	no
Matrices / Complex	hadrumaths	library	yes	library	library	yes
Control structures overloading (redefine for/while)	yes	no	no	no	no	no
Single file project (with dependencies)	yes	no	Comp. opt.	Comp. Opt.	yes	Mex or javaaddpath
Paradigm	func/OO	OO	Proc	OO/Proc	OO	Proc, supports OO
BLAS and LAPACK	hadrumaths	JBLAS library	Seamless integration	Library integration	Library integration	MEX file + Library
Elastic Numeric Calculation	yes	no	no	no	no	no

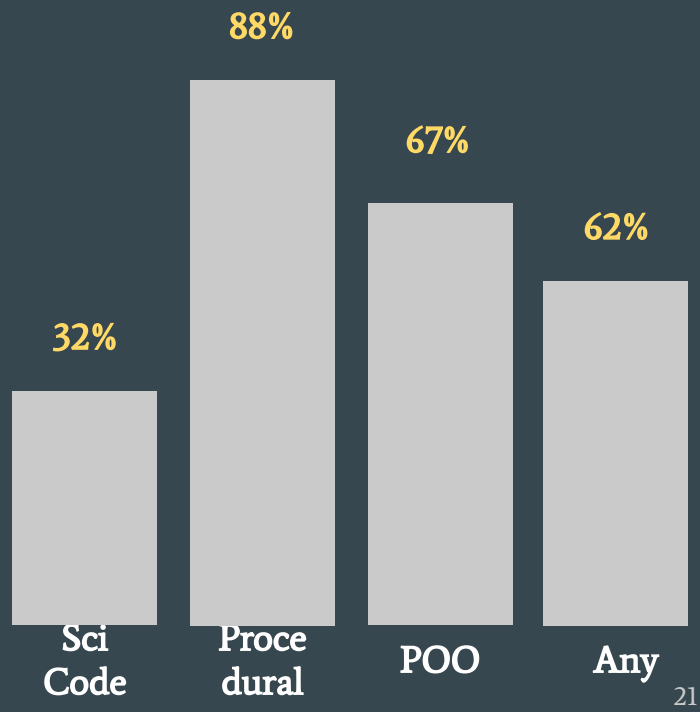
Performance

- Current version of Hadra generates Java sources then compiles to ByteCode (for validation purposes)
- Small Performance enhancements due to literal optimizations
  - Regexp
  - Dates
  - Primitive types
- Performance tested against
  - https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html
  - Considered: the best results



Conciseness

- Comparing Hadra source length to Java equivalent code
- Using sample code
  - #1: Sci code: using operator overloading in hadrumaths
  - #2: Java purely procedural
  - #3 Data classes,Typical Java code
  - #4 average of all the above



Why Hadra

1	2	3

• First OO Programming Language is “Small Talk”
• Hadra means “Lots of talk” in Tunisian
• It means also “Interesting Thing”
	• Hadra builds upon existing work and libraries in the Laboratory:
	  ○ Hadrumaths
	  ○ Hadruwaves
	  ○ Hadruplots
		• The prefix hadru and the name hadra come from “Hadrumet”, the Phoenician name of Sousse, the city where the authors are from

Conclusion

Existing Sci Languages	• DSL / GPL Blurry boundaries • No clear winner
Proposed a new Language	• Readable, Concise, Simple • Based on JVM: portable
TODO: Tooling	• Under construction, Netbeans/Intellij Integ. • Syntax Highlighting in Kate/Sublime etc.
TODO: Sources & Perf	• To be published shortly under OSS License • (now as private GITHUB repository)

GRAALVM

Thank you

taha.bensalah@gmail.com
taha.bensalah@eniso.u-sousse.tn
http://github.com/thevpc