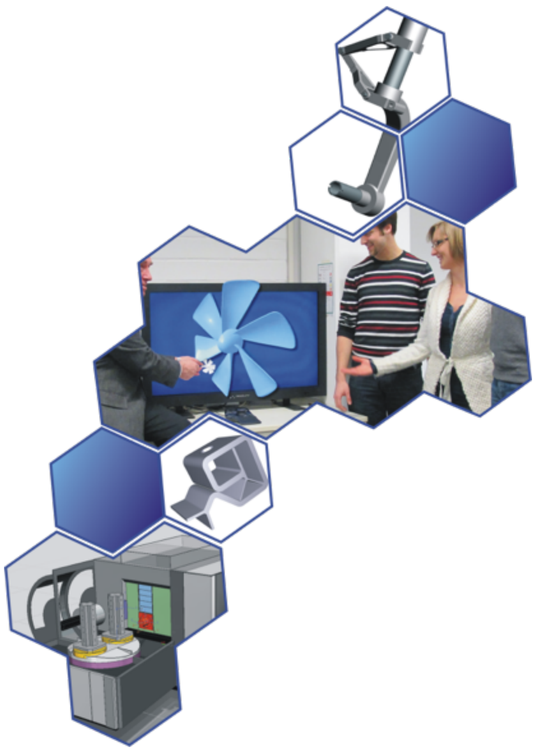
A Comparative Study of Programming Languages for Next-Generation Astrodynamics Systems

Helge Eichhorn, Juan Luis Cano, Frazer McLean,
Prof Dr.-Ing. Reiner Anderl



Speed/Usability Dichotomy

1970

Computing costs
are the limiting factor

Speed/Usability Dichotomy

1970
Computing costs
are the limiting factor

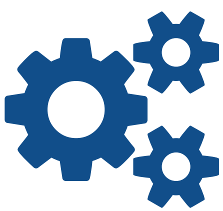
Efficiency is most
important

Speed/Usability Dichotomy

TECHNISCHE UNIVERSITÄT DARMSTADT

1970
Computing costs
are the limiting factor

2016
Personnel costs
are the limiting factor



Efficiency is most important

Speed/Usability Dichotomy

TECHNISCHE UNIVERSITÄT DARMSTADT

1970
Computing costs
are the limiting factor

2016
Personnel costs
are the limiting factor

Efficiency is most important

Usability should be most important

Speed/Usability Dichotomy

1970
Computing costs
are the limiting factor

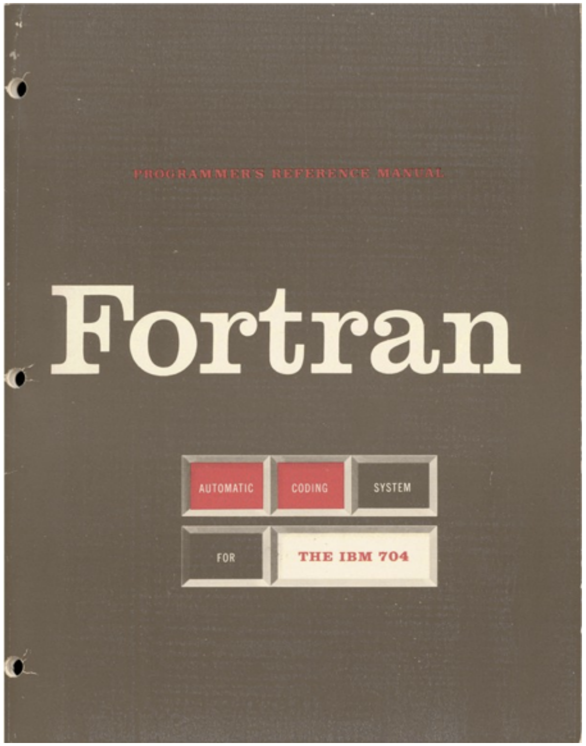
2016
Personnel costs
are the limiting factor

Efficiency is most
important

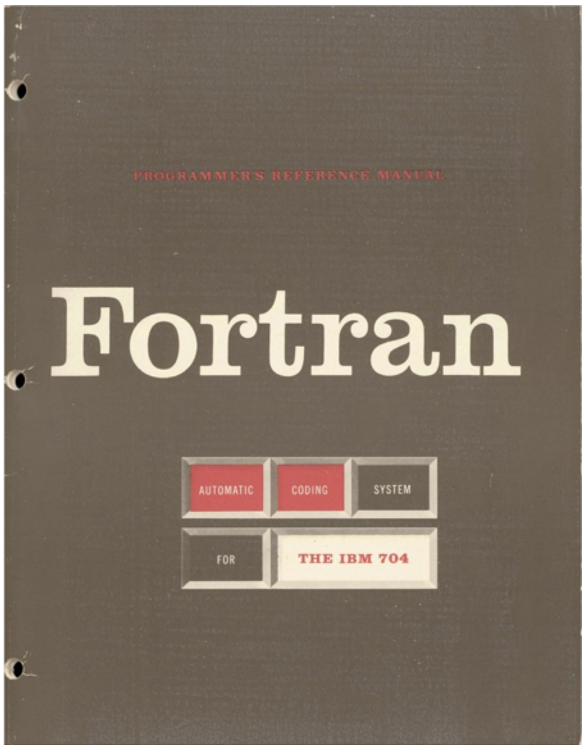
Cannot have both?

Usability should
be most important

Fortran

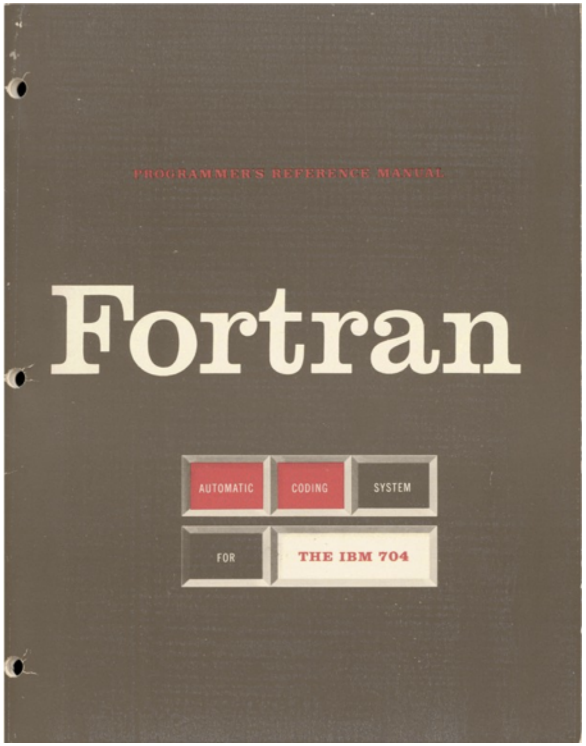

PROGRAMMER'S REFERENCE MANUAL
Fortran
AUTOMATIC
CODING
SYSTEM
FOR
THE IBM 704

Fortran



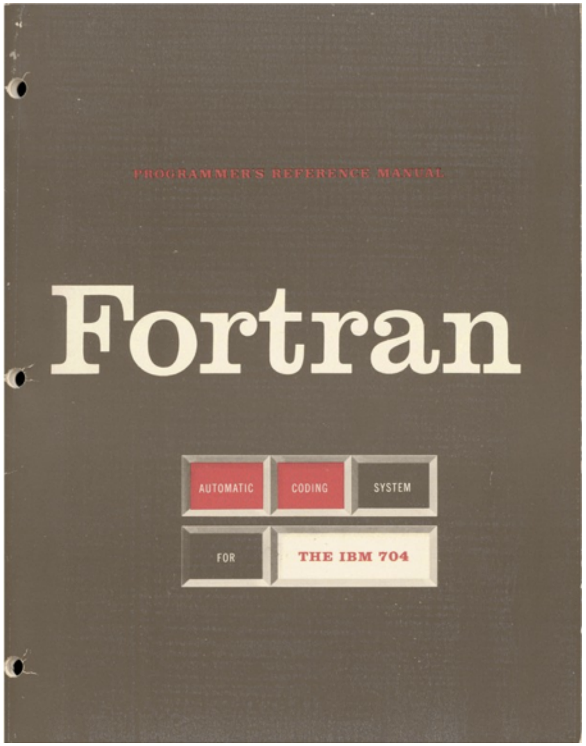
- Tried and tested

Fortran



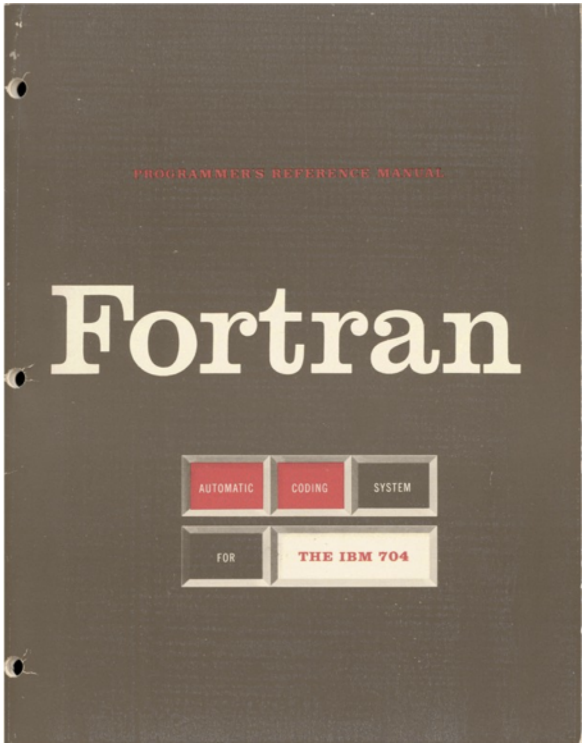
- Tried and tested
- Fast

Fortran



- Tried and tested
- Fast
- Fortran 90+ offers great improvements…

Fortran



- Tried and tested
- Fast
- Fortran 90+ offers great improvements…
- …but also increases complexity.

C++

C++

TECHNISCHE UNIVERSITÄT DARMSTADT

- Powerful and versatile

- Powerful and versatile
- Fast

C++

- Powerful and versatile
- Fast
- Complex and difficult to master

C++

- Powerful and versatile
- Fast
- Complex and difficult to master
- No training wheels

Java

Java

- Mature toolchain

Java

- Mature toolchain
- Large community and ecosystem

Java

- Mature toolchain
- Large community and ecosystem
- Language of Big Data

Java

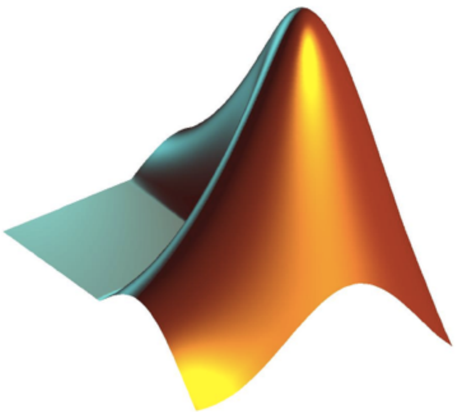
- Mature toolchain
- Large community and ecosystem
- Language of Big Data
- Class-based OOP is not a panacea

Matlab

Matlab

- Easy to learn

Matlab

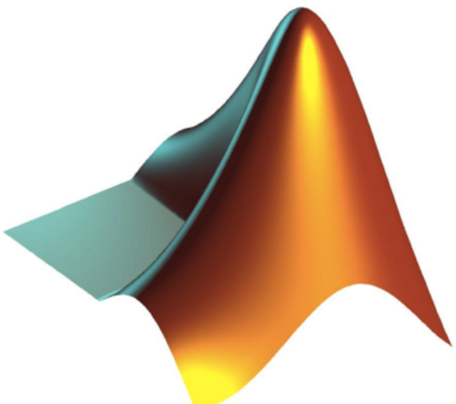


- Easy to learn
- Powerful environment

Matlab

- Easy to learn
- Powerful environment
- Expensive

Matlab



- Easy to learn
- Powerful environment
- Expensive
- Core language is limited

Python

Python

- Easy to learn

Python

- Easy to learn
- Batteries included

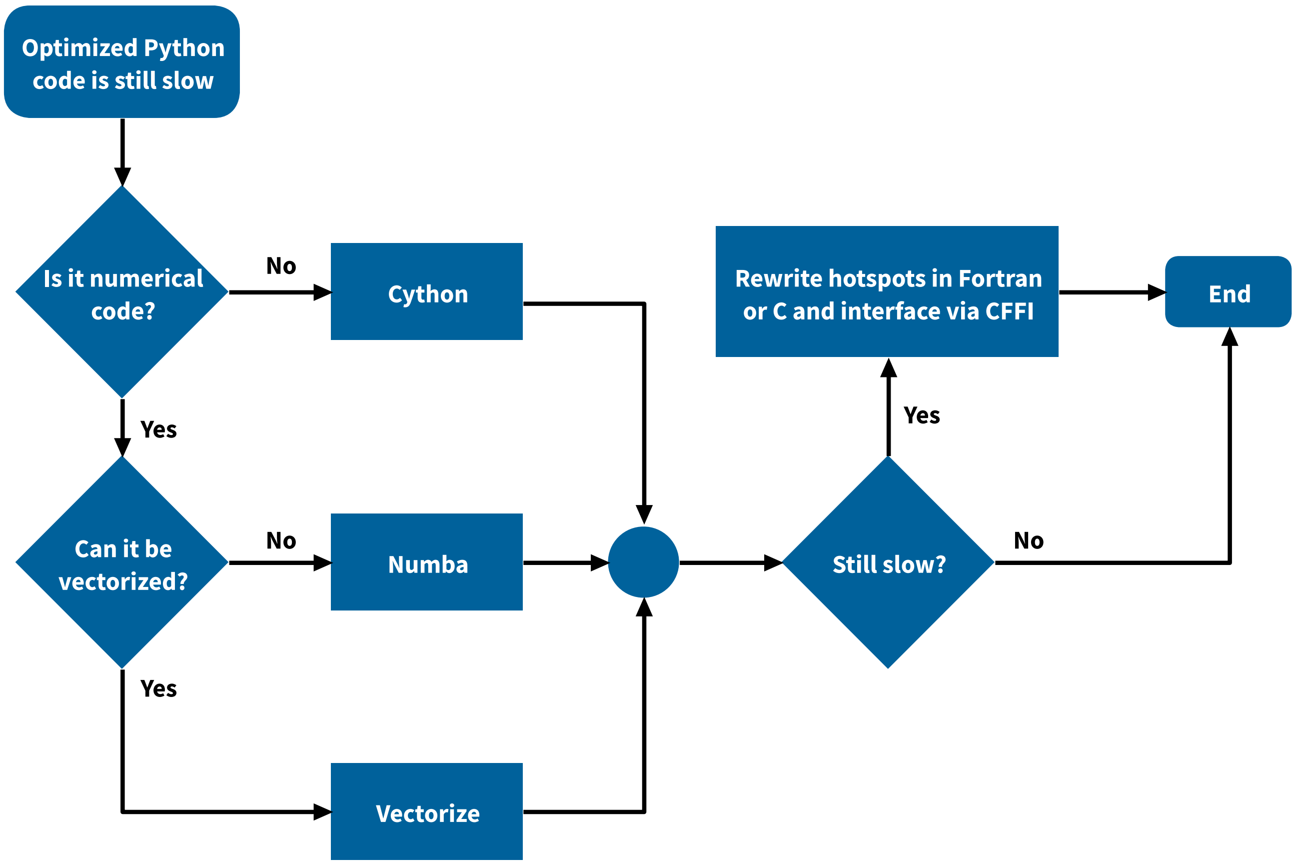
Python

- Easy to learn
- Batteries included
- Large scientific computing ecosystem

Python

- Easy to learn
- Batteries included
- Large scientific computing ecosystem
- (Too) many optimization options

Python Optimization



Julia

Julia

- Matlab-like syntax

Julia

- Matlab-like syntax
- Fast

Julia

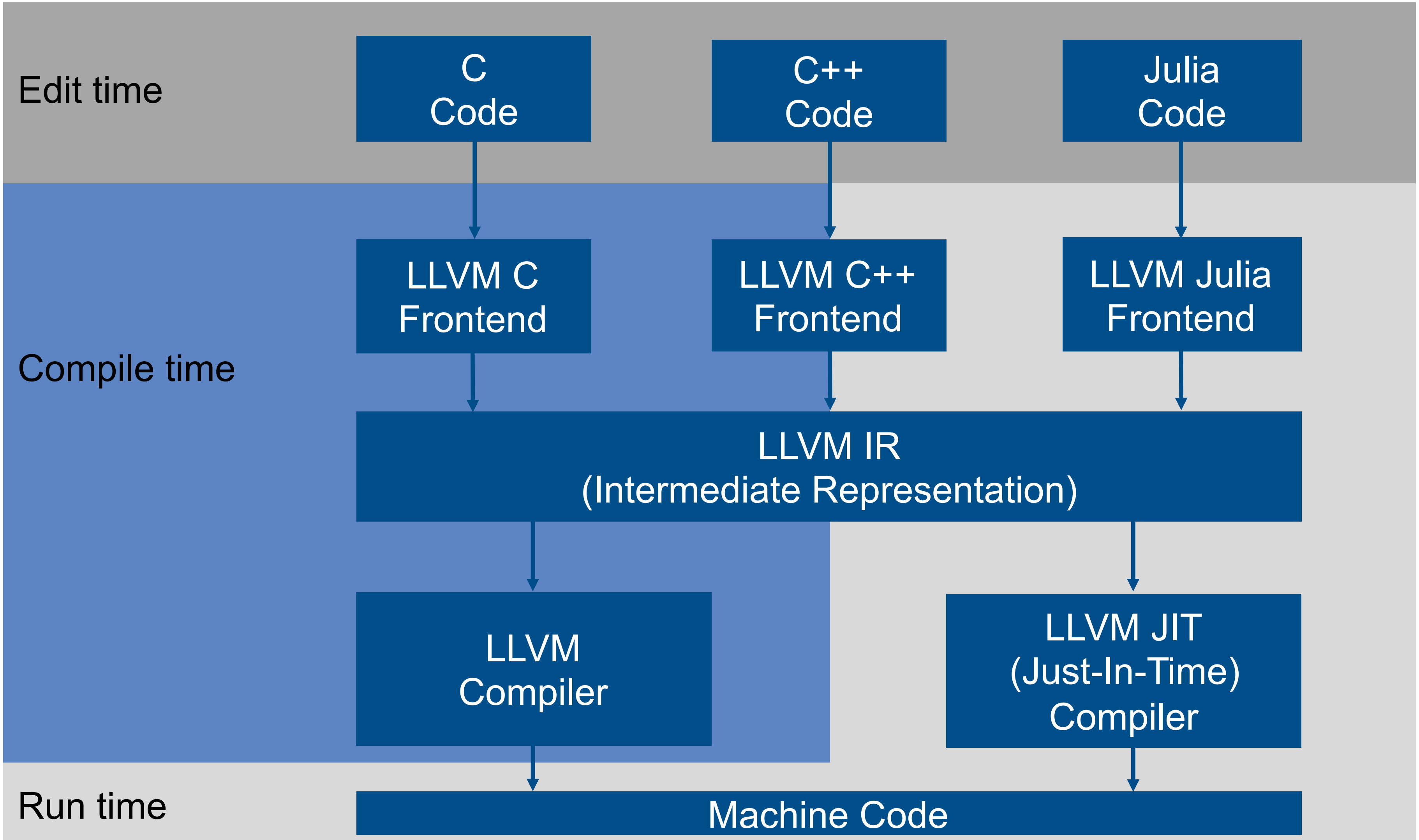
- Matlab-like syntax
- Fast
- Multiple dispatch

Julia

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Matlab-like syntax
- Fast
- Multiple dispatch
- Immature

LLVM


Edit time
C Code
C++ Code
Julia Code
Compile time
LLVM C Frontend
LLVM C++ Frontend
LLVM Julia Frontend
LLVM IR
(Intermediate Representation)
LLVM Compiler
LLVM JIT
(Just-In-Time)
Compiler
Run time
Machine Code

Test Cases

1. Calculating the Keplerian orbital elements

Test Cases

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Calculating the Keplerian orbital elements

2. Solving Kepler's equation

Test Cases

1. Calculating the Keplerian orbital elements

2. Solving Kepler's equation

3. Solving Lambert's problem

Test Cases

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Calculating the Keplerian orbital elements
2. Solving Kepler's equation
3. Solving Lambert's problem
4. Calling the DOP853 Fortran 77 code

Test 1: Keplerian Elements

TECHNISCHE UNIVERSITÄT DARMSTADT

How well can vector expressions be expressed?

How well can vector expressions be expressed?

Julia

```
e = ((v_mag^2 - μ/r_mag)*r - (r⋅v)*v)/μ
```

Test 1: Keplerian Elements

How well can vector expressions be expressed?

Julia

```
e = ((v_mag^2 - μ/r_mag)*r - (r⋅v)*v)/μ
```

Java

```
e = new Vector3D(v_mag*v_mag / mu -
1/r_mag, r, -r.dotProduct(v) / mu, v);
```

Test 2: Kepler’s Equation

Can functions be created ad-hoc (higher-order functions) and do they have access to their enclosing scope (closures)?

Test 2: Kepler's Equation

Can functions be created ad-hoc (higher-order functions) and do they have access to their enclosing scope (closures)?

Yes.

Test 2: Kepler's Equation

Can functions be created ad-hoc (higher-order functions) and do they have access to their enclosing scope (closures)?

Yes.

Except for Fortran.

Test 2: Kepler’s Equation

Can functions be created ad-hoc (higher-order functions) and do they have access to their enclosing scope (closures)?

Yes.

Except for Fortran.

```
def mean2ecc(M, ecc):
    def keplereq(E):
        return E - ecc*np.sin(E) - M
    def keplerderiv(E):
        return 1 - ecc*np.cos(E)
    return newton(M, keplereq, keplerderiv)
```

Test 2: Kepler's Equation

Can functions be created ad-hoc (higher-order functions) and do they have access to their enclosing scope (closures)?

Yes.

Except for Fortran.

```
def mean2ecc(M, ecc):
    def keplereq(E):
        return E - ecc*np.sin(E) - M
    def keplerderiv(E):
        return 1 - ecc*np.cos(E)
    return newton(M, keplereq, keplerderiv)
```

Test 3: Lambert's Problem ➤ Performance test

Test 4: Interfacing with Fortran 77

How much additional glue code is required to call a Fortran77 subroutine? Can the Fortran code call back?

Test 4: Interfacing with Fortran 77

TECHNISCHE UNIVERSITÄT DARMSTADT

How much additional glue code is required to call a Fortran77 subroutine? Can the Fortran code call back?

- Fortran2008, C++, Julia: No glue code required. Callbacks are possible.

Test 4: Interfacing with Fortran 77

How much additional glue code is required to call a Fortran77 subroutine? Can the Fortran code call back?

- Fortran2008, C++, Julia: No glue code required. Callbacks are possible.
- Python: Moderate amounts of glue code required. Callbacks are possible.

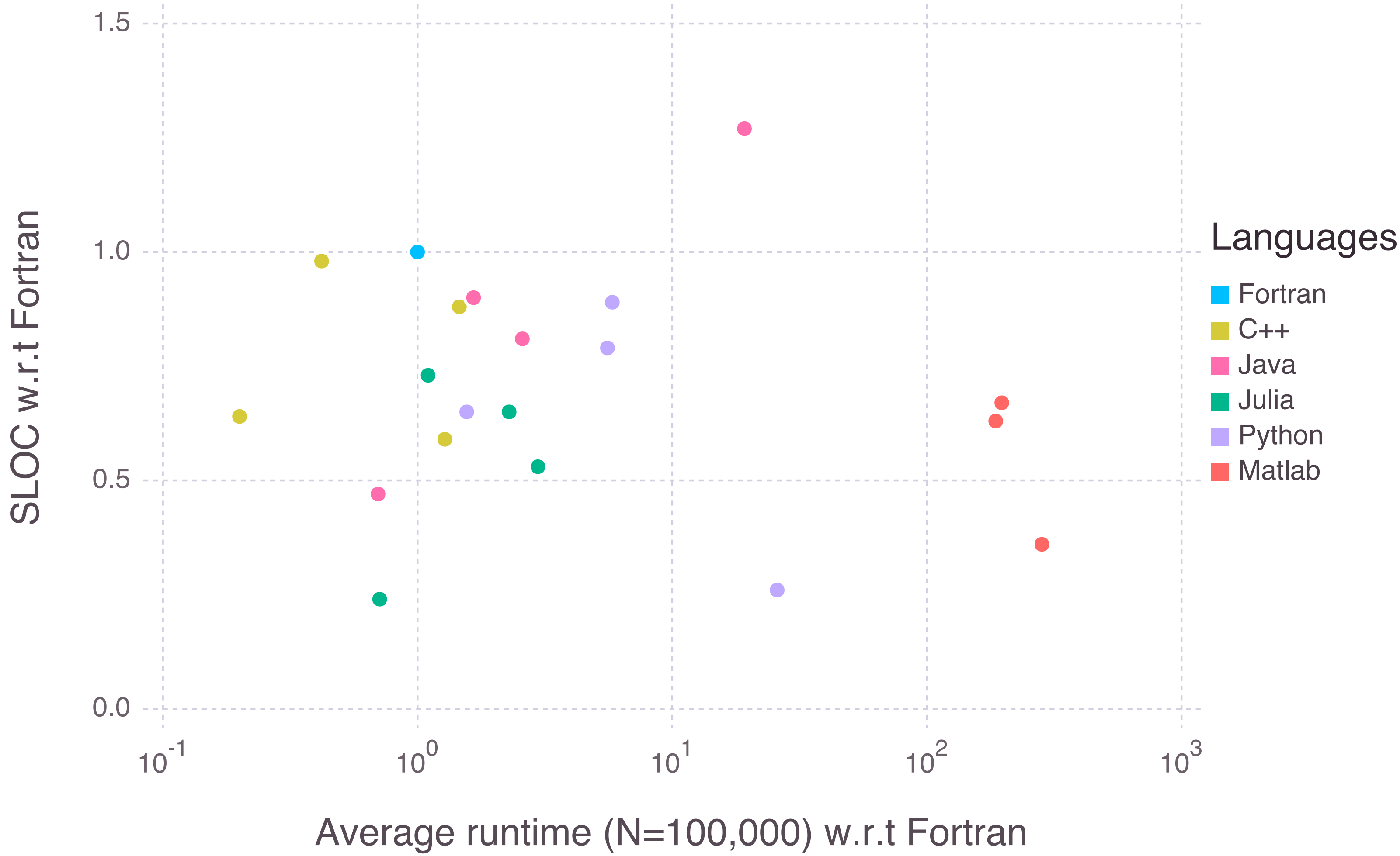
Test 4: Interfacing with Fortran 77

How much additional glue code is required to call a Fortran77 subroutine? Can the Fortran code call back?

- Fortran2008, C++, Julia: No glue code required. Callbacks are possible.
- Python: Moderate amounts of glue code required. Callbacks are possible.
- Java, Matlab: Larger amounts of glue code required. Callbacks might require changes to the Fortran code.

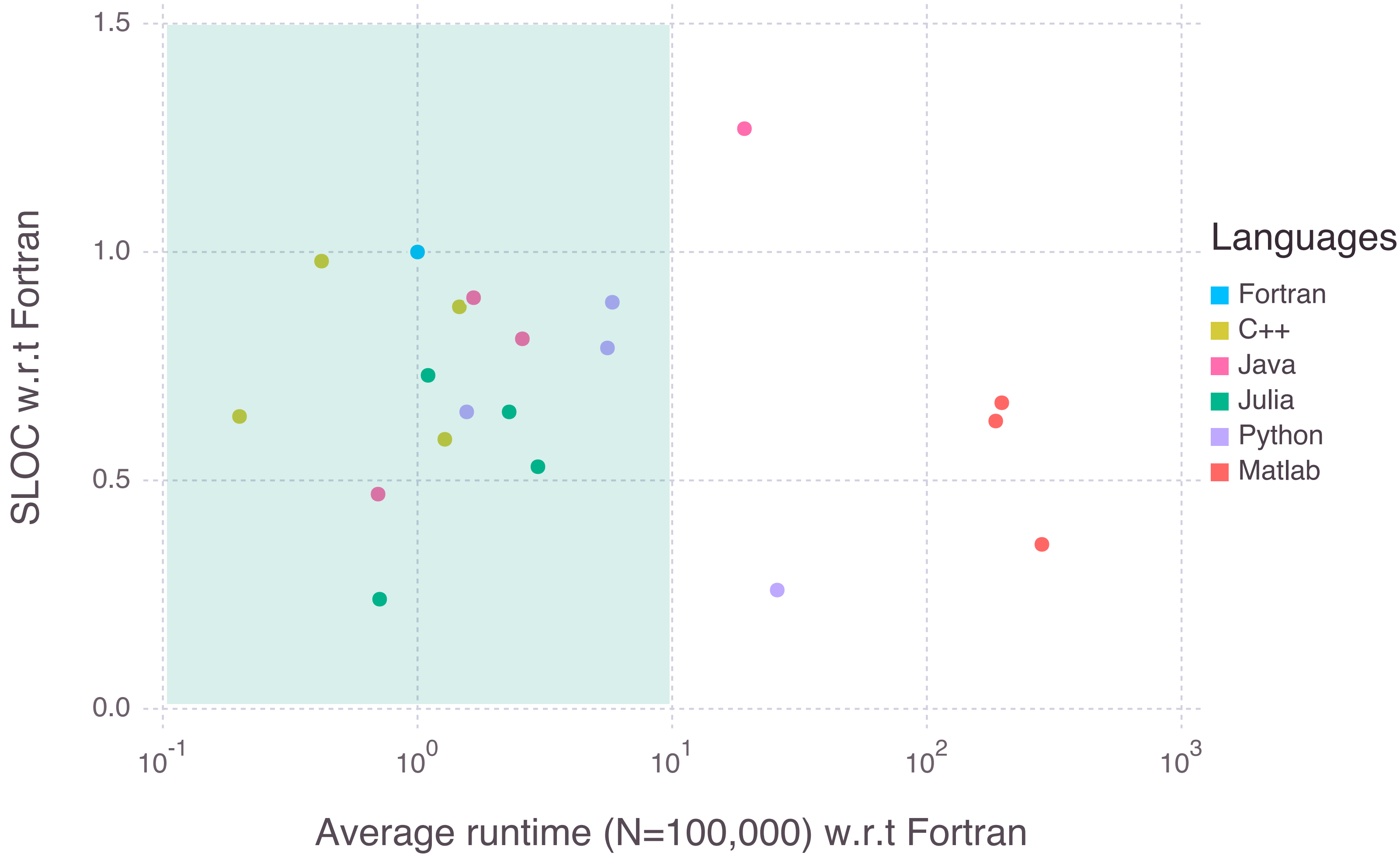
Benchmark

https://github.com/helgee/icatt-2016



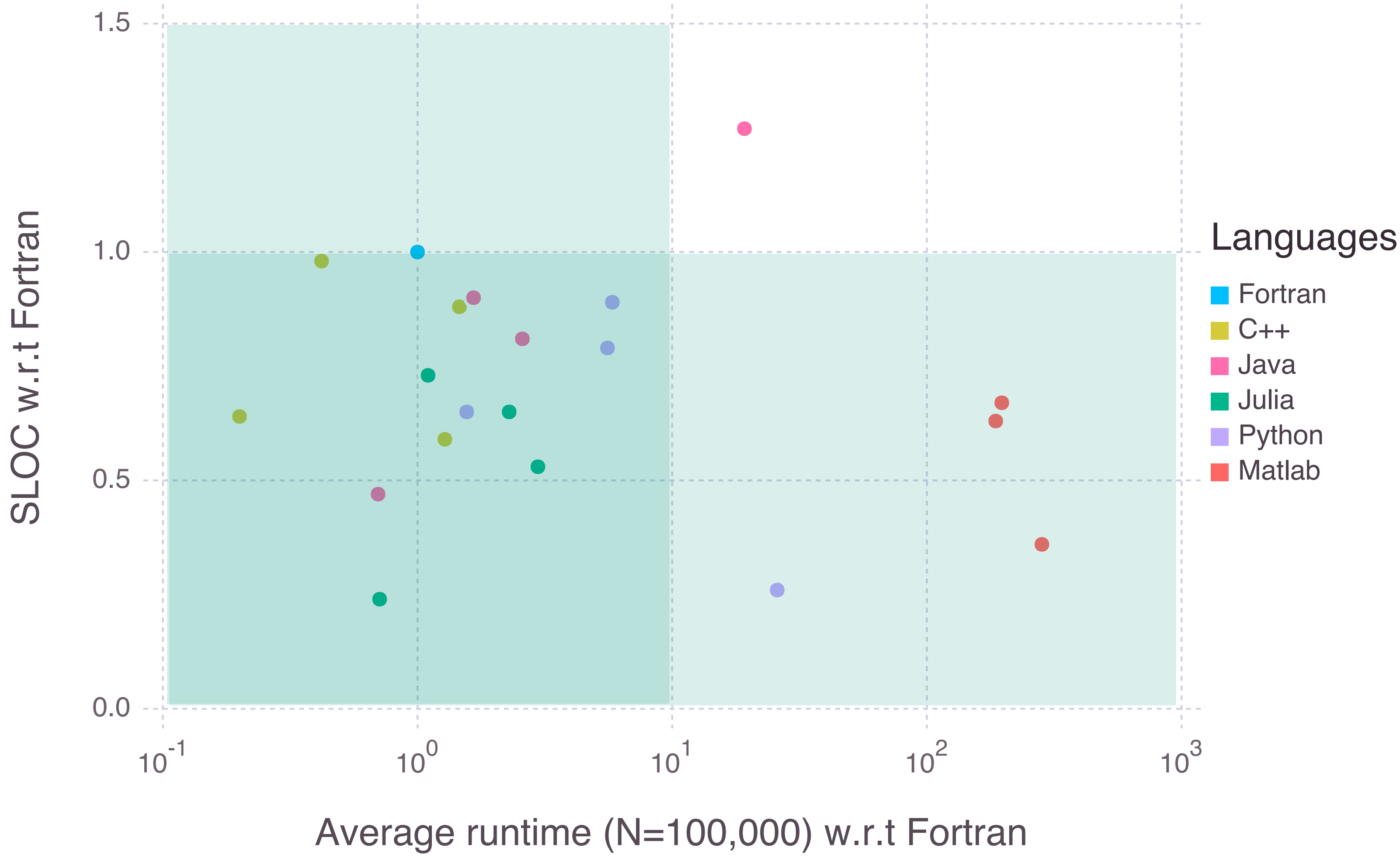
Benchmark

https://github.com/helgee/icatt-2016



Benchmark

https://github.com/helgee/icatt-2016



Conclusion

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Statically compiled languages have borrowed features from dynamic languages and still offer best-in-class performance.

Conclusion

- Statically compiled languages have borrowed features from dynamic languages and still offer best-in-class performance.

- Purely interpreted languages remain orders of magnitude slower but JIT-compiled dynamic languages have become competitive.

Conclusion

- Statically compiled languages have borrowed features from dynamic languages and still offer best-in-class performance.
- Purely interpreted languages remain orders of magnitude slower but JIT-compiled dynamic languages have become competitive.
- Python+Numba and Julia offer an attractive compromise between flexibility and performance.