BNumMet

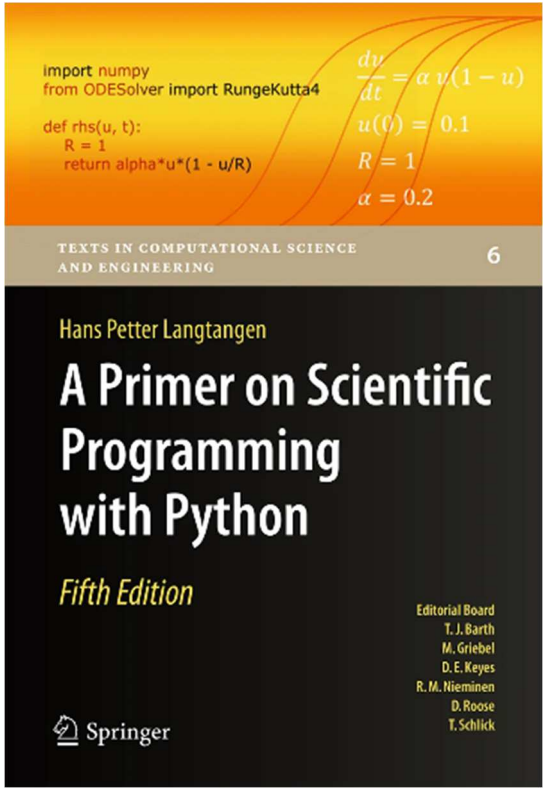
A Scholar implementation of Numerical Methods in Python enhanced with interactive widgets

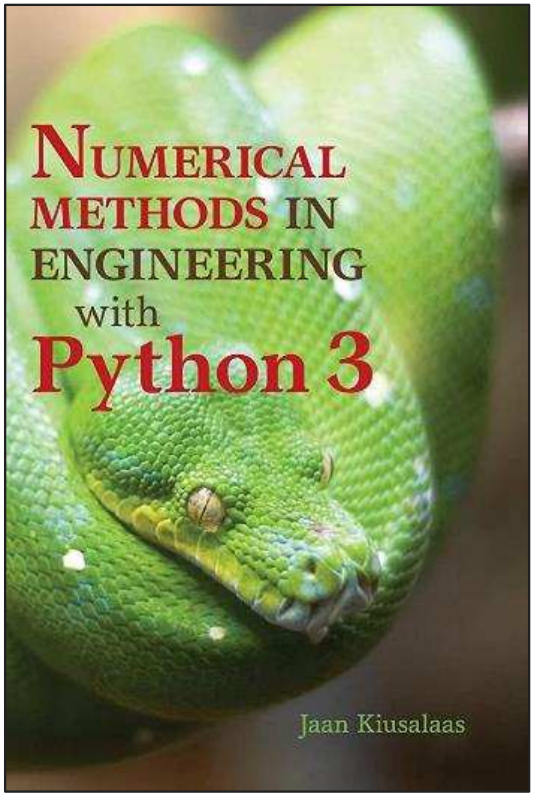
Author: Fernando Bellido Pazos

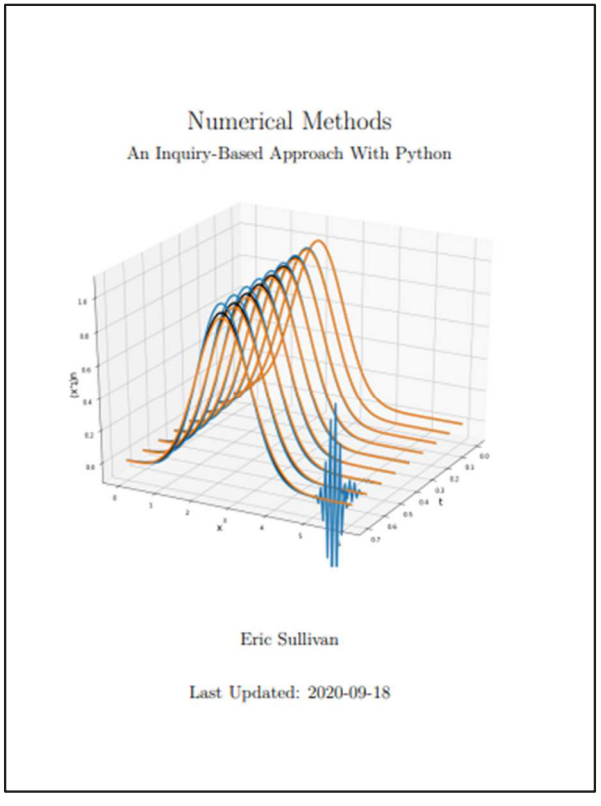
Tutor: Juan Manuel Perez Pardo

Contents

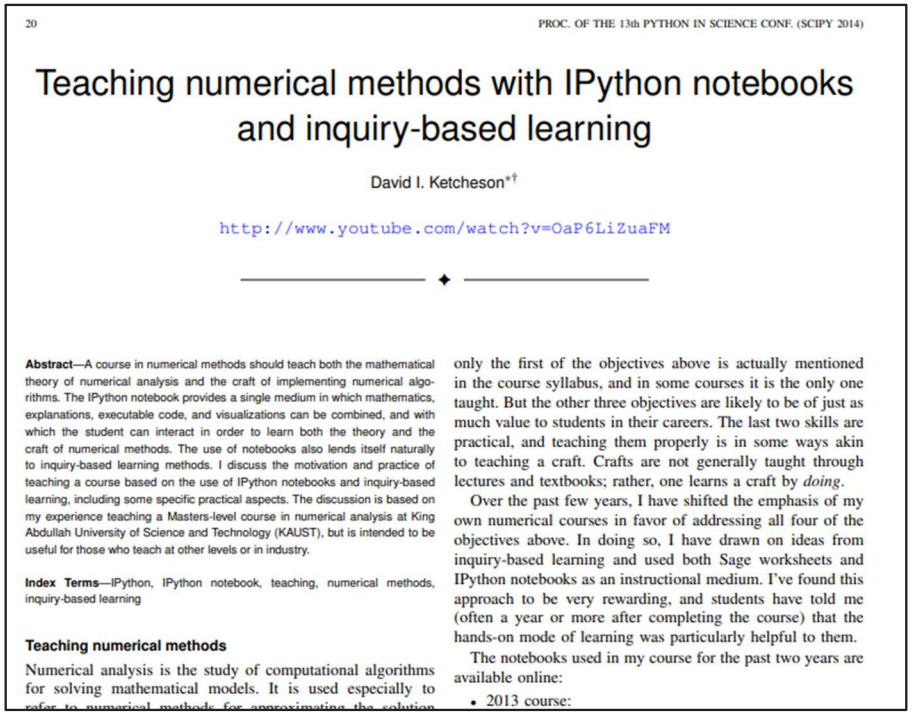
State of the Art







State of the Art



20 PROC. OF THE 13th PYTHON IN SCIENCE CONF. (SCIPY 2014)

Teaching numerical methods with IPython notebooks and inquiry-based learning

David I. Ketcheson*†

http://www.youtube.com/watch?v=0aP6LiZuaFM

Abstract—A course in numerical methods should teach both the mathematical theory of numerical analysis and the craft of implementing numerical algorithms. The IPython notebook provides a single medium in which mathematics, explanations, executable code, and visualizations can be combined, and with which the student can interact in order to learn both the theory and the craft of numerical methods. The use of notebooks also lends itself naturally to inquiry-based learning methods. I discuss the motivation and practice of teaching a course based on the use of IPython notebooks and inquiry-based learning, including some specific practical aspects. The discussion is based on my experience teaching a Masters-level course in numerical analysis at King Abdullah University of Science and Technology (KAUST), but is intended to be useful for those who teach at other levels or in industry.

Index Terms—IPython, IPython notebook, teaching, numerical methods, inquiry-based learning

Teaching numerical methods

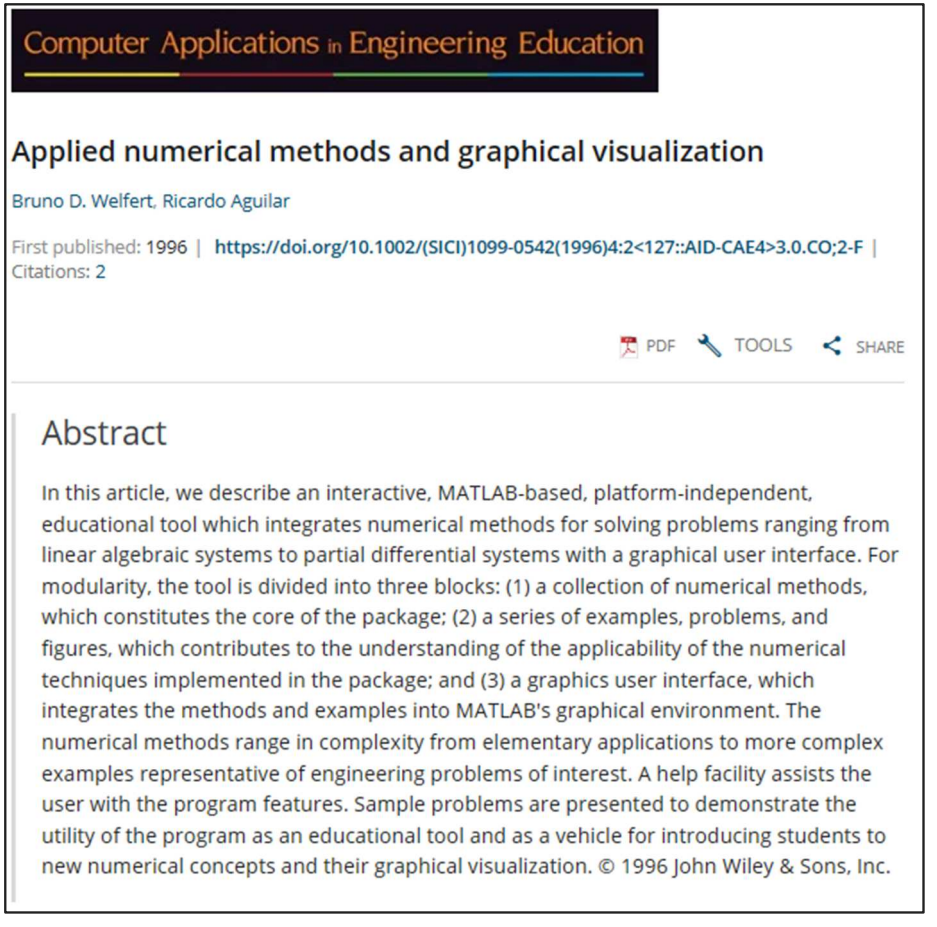
Numerical analysis is the study of computational algorithms for solving mathematical models. It is used especially to refer to numerical methods for approximating the solution

only the first of the objectives above is actually mentioned in the course syllabus, and in some courses it is the only one taught. But the other three objectives are likely to be of just as much value to students in their careers. The last two skills are practical, and teaching them properly is in some ways akin to teaching a craft. Crafts are not generally taught through lectures and textbooks; rather, one learns a craft by *doing*.

Over the past few years, I have shifted the emphasis of my own numerical courses in favor of addressing all four of the objectives above. In doing so, I have drawn on ideas from inquiry-based learning and used both Sage worksheets and IPython notebooks as an instructional medium. I've found this approach to be very rewarding, and students have told me (often a year or more after completing the course) that the hands-on mode of learning was particularly helpful to them.

The notebooks used in my course for the past two years are available online:

- 2013 course:



Computer Applications in Engineering Education

Applied numerical methods and graphical visualization

Bruno D. Welfert, Ricardo Aguilar

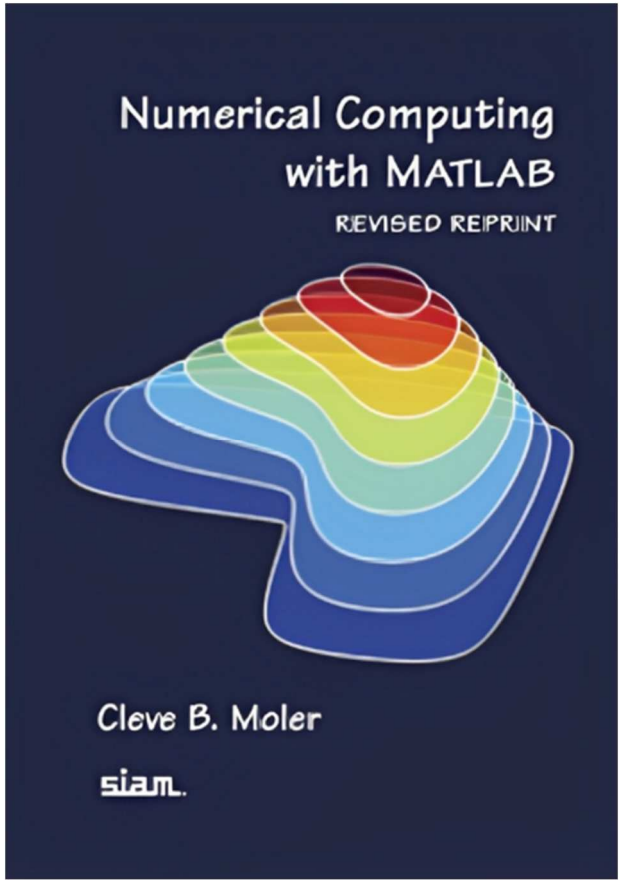
First published: 1996 | https://doi.org/10.1002/(SICI)1099-0542(1996)4:2<127::AID-CAE4>3.0.CO;2-F |
Citations: 2

PDF TOOLS SHARE

Abstract

In this article, we describe an interactive, MATLAB-based, platform-independent, educational tool which integrates numerical methods for solving problems ranging from linear algebraic systems to partial differential systems with a graphical user interface. For modularity, the tool is divided into three blocks: (1) a collection of numerical methods, which constitutes the core of the package; (2) a series of examples, problems, and figures, which contributes to the understanding of the applicability of the numerical techniques implemented in the package; and (3) a graphics user interface, which integrates the methods and examples into MATLAB's graphical environment. The numerical methods range in complexity from elementary applications to more complex examples representative of engineering problems of interest. A help facility assists the user with the program features. Sample problems are presented to demonstrate the utility of the program as an educational tool and as a vehicle for introducing students to new numerical concepts and their graphical visualization.

Previous Work



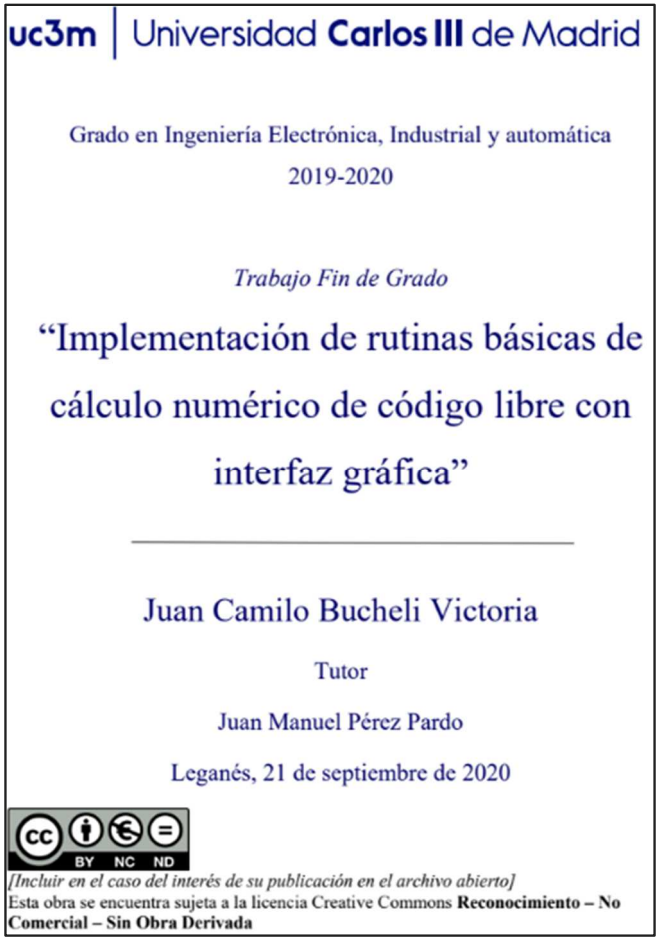
Numerical Computing With MATLAB – Cleve B.Moler

- Programming Language: MATLAB
- 11 Chapters : Code & Visualization
- Private Licensed
- Graphical User Interface: Not Intuitive and user-friendly
- MATLAB's Home License: 119€

Previous Work

End of Degree's Thesis – Juan Camilo Bucheli

- Programming Language: Python
- 2 Packages: Linear Systems & Interpolation
- Open Source Licensed (Apache 2.0)
- Graphical User Interface: NOT functional to this day
- Language: Spanish



uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Electrónica, Industrial y automática

2019-2020

Trabajo Fin de Grado

"Implementación de rutinas básicas de cálculo numérico de código libre con interfaz gráfica"

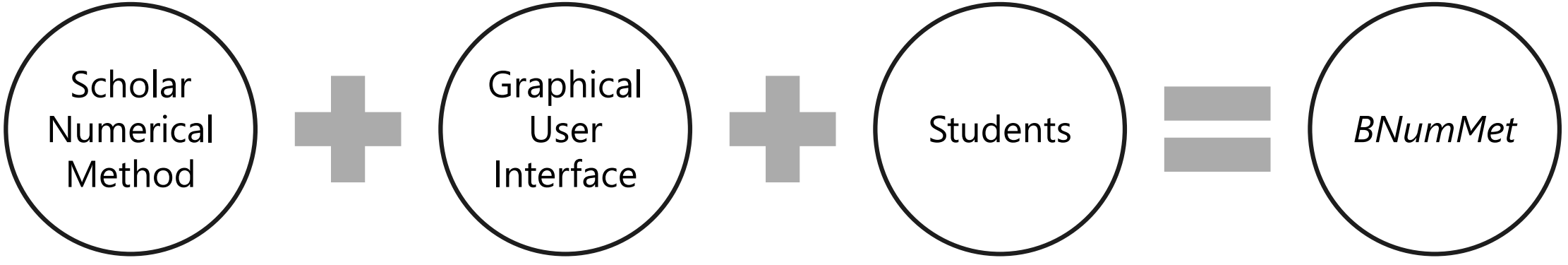
Juan Camilo Bucheli Victoria

Tutor

Juan Manuel Pérez Pardo

Leganés, 21 de septiembre de 2020

Objectives


Scholar Numerical Method
Graphical User Interface
Students
BNumMet

Objectives

General Code

- [ ] Open Source
- [ ] English
- [ ] Easy to Read
- [ ] Self-Contained
- [ ] Efficiency
- [ ] Functional

G.U.I.

- [ ] Provide Insight on algorithm
- [ ] Interact and observe changes Dynamically
- [ ] Easy to use
- [ ] Modular on Input
- [ ] PC-Human Interaction theory

Software Development

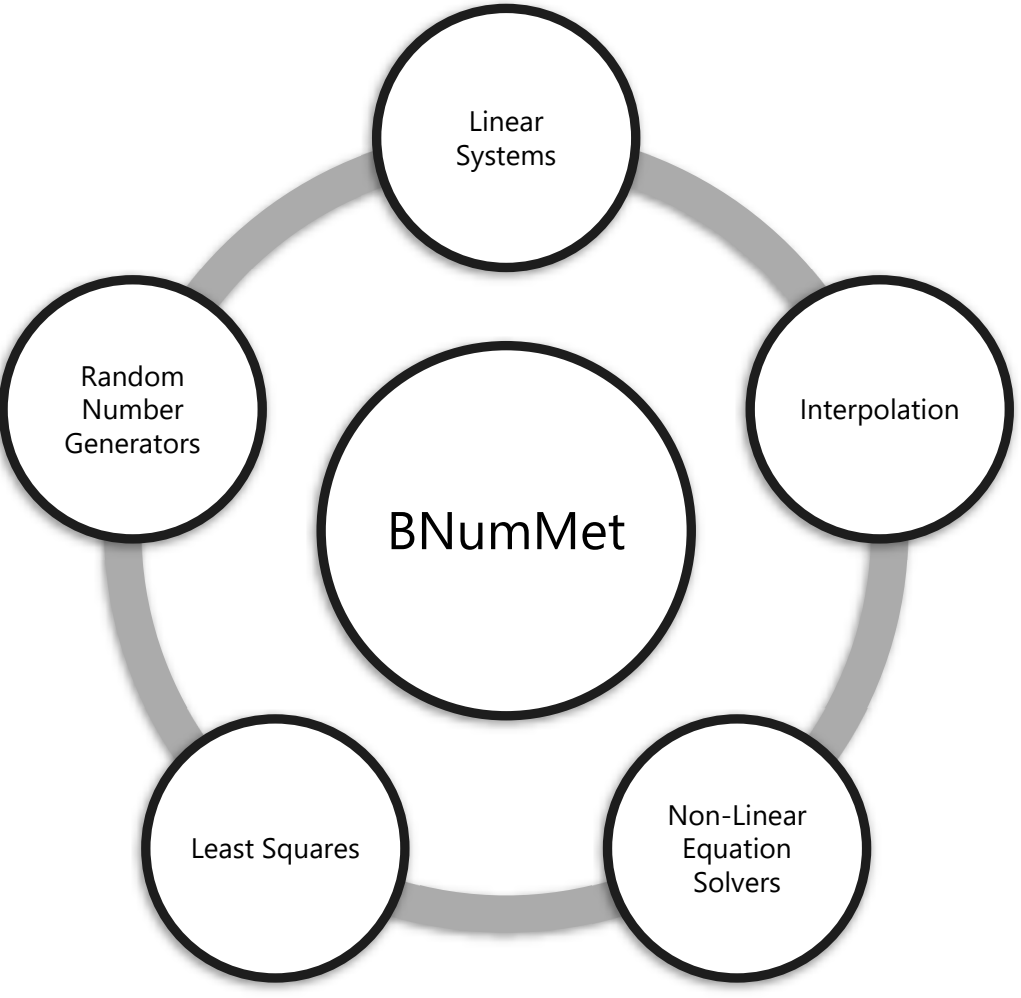
- **Roles**
- **Life Cycle Model:** Iterative Life Cycle
- **Development Framework:** Python ≥ 3.8 & Jupyter
- **Licensing:** GNU Affero General Public Licence v3.0

- **Schedule Frame:** October 2022 - 15 June 2023 in 2 Week Iterations

- **Quality Assurance:** Unit Test & SonarQube
- **Software Configuration Management:** Github & vX.Y.Z Scheme

- **Packaging:** PIP & Public Github

Software Description



- Written from scratch
- Original Implementations
- MathWorks Benchmarking
- Analysis of Solutions provided

Linear Systems

Packages

LU Decomposition*

Forward & Backward Substitution

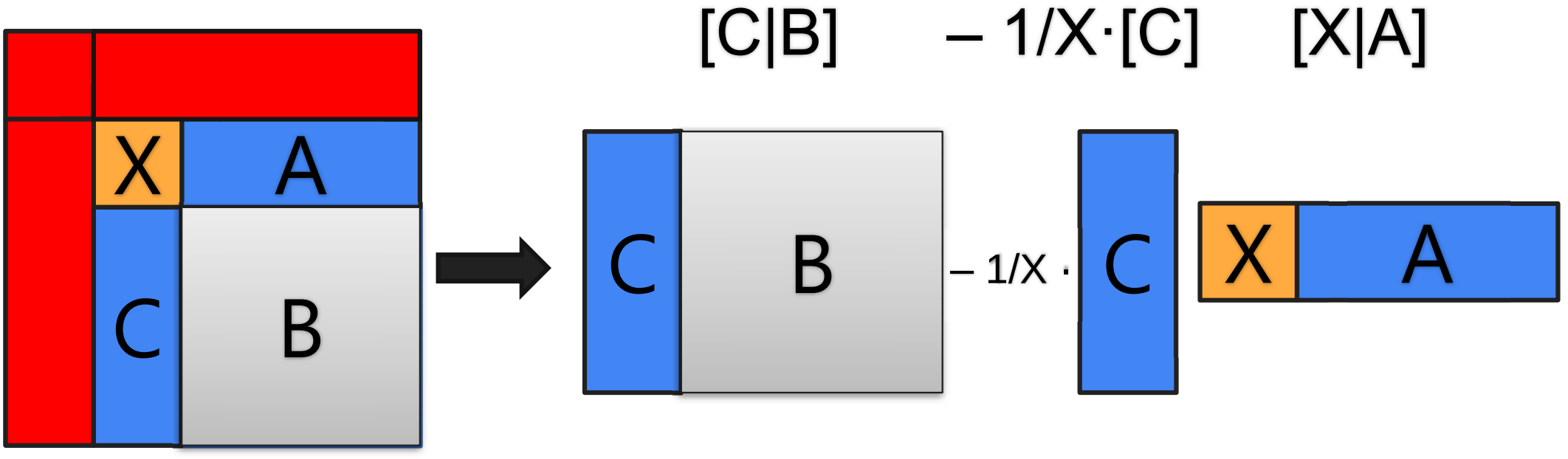
LU Solver

QR Decomposition

QR Solver

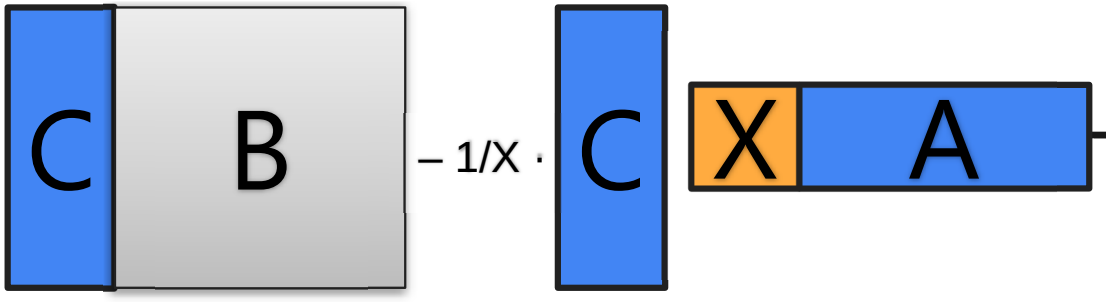
Linear Systems

Discussion: LU Decomposition



Linear Systems

Discussion: LU Decomposition



Submatrices Syntax

```
A[col + 1 :, :][:, [col]] @ A[[col], :][:, col + 1 :]
```

New-Axis Syntax

```
A[col + 1 :, col][:, np.newaxis]
@ A[col, col + 1 :][np.newaxis, :]
```

Outer Product

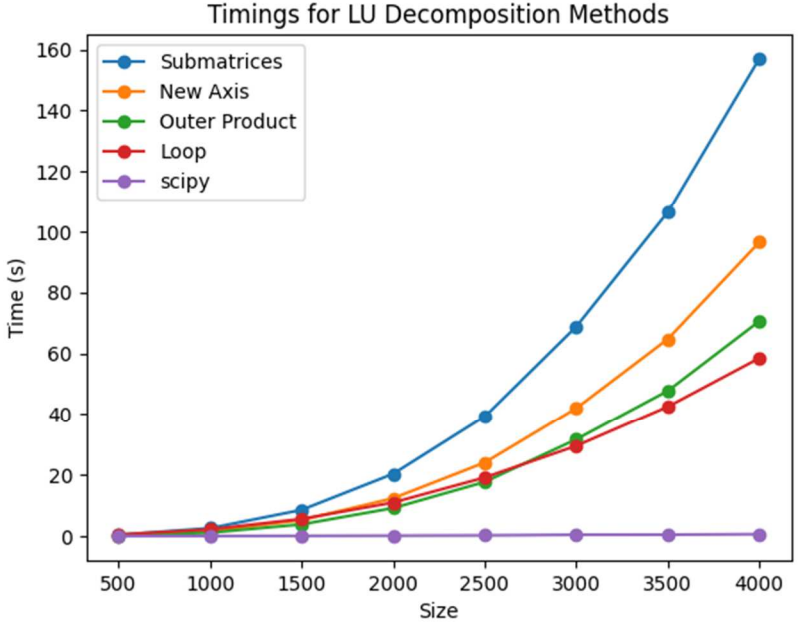
```
np.outer(A[col + 1 :, col], A[col, col + 1 :])
```

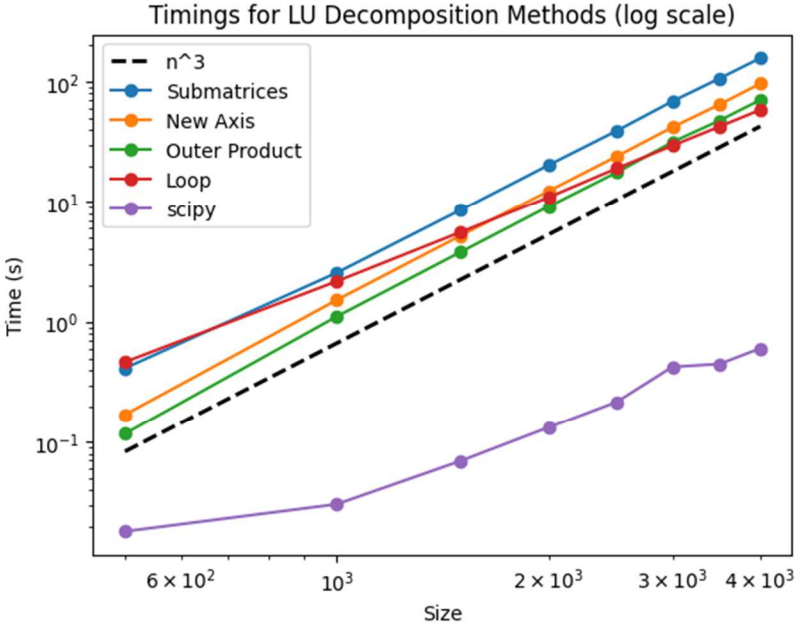
For Loop

```
for col in range(n):
<Pivoting>
for row in range(col + 1, n):
A[row, col + 1 :] - A[row, col] * A[col, col + 1 :]
```

Linear Systems

Discussion: LU Decomposition





	Slope
Submatrices	2.8852
New Axis	3.0434
Outer Product	3.0717
Loop	2.334
Scipy	1.8136

Interpolation

Packages

Polynomial Interpolation

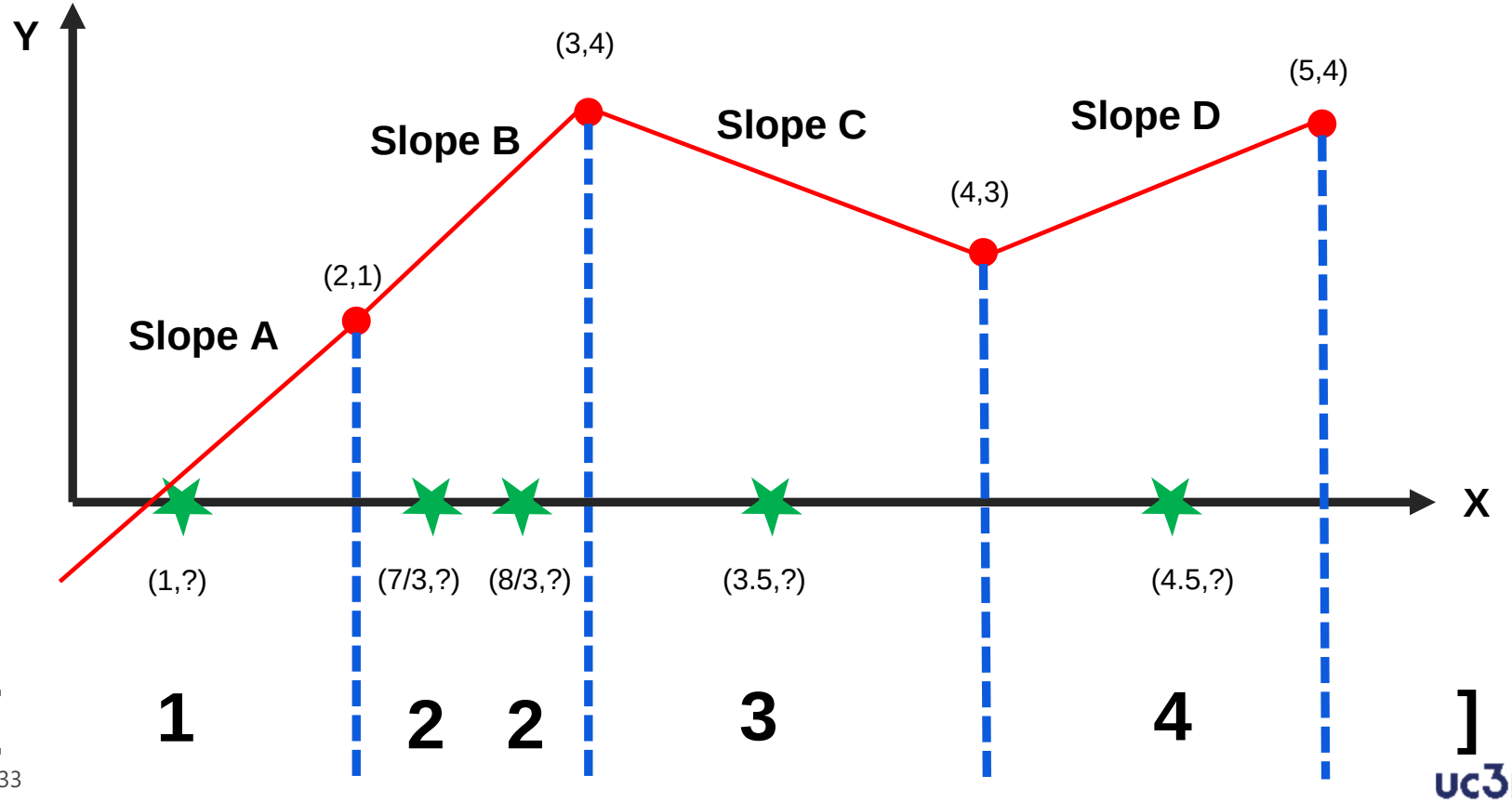
Piecewise Linear Interpolation

Piecewise Cubic Interpolation

Piecewise Cubic Hermite Interpolation

Interpolation

Discussion: Interpolation



Pages 31-33

Interpolation

Discussion: Interpolation

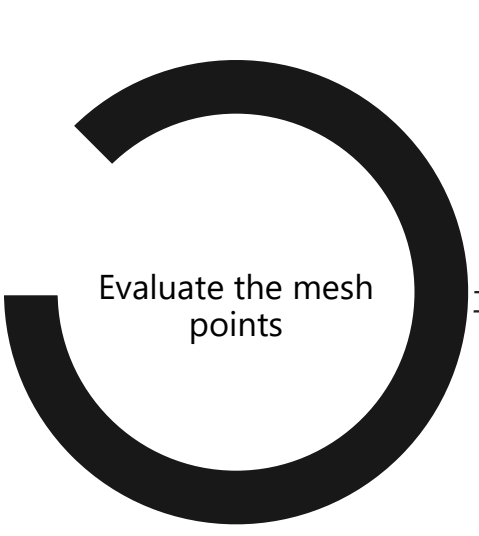
K=[

1	(1,?)	Evaluate using slope A
2	(7/3,?)	Evaluate using slope B
2	(8/3,?)	
3	(3.5,?)	Evaluate using slope C
4	(4.5,?)	Evaluate using slope C

]

Interpolation

Discussion: Interpolation



List Comprehension

```
v = [ y[k[i] - 1] + s[i] * (d[k[i] - 1] + s[i] * (c[k[i] - 1] + s[i] * b[k[i] - 1]))
      for i in range(len(u)) ]
```

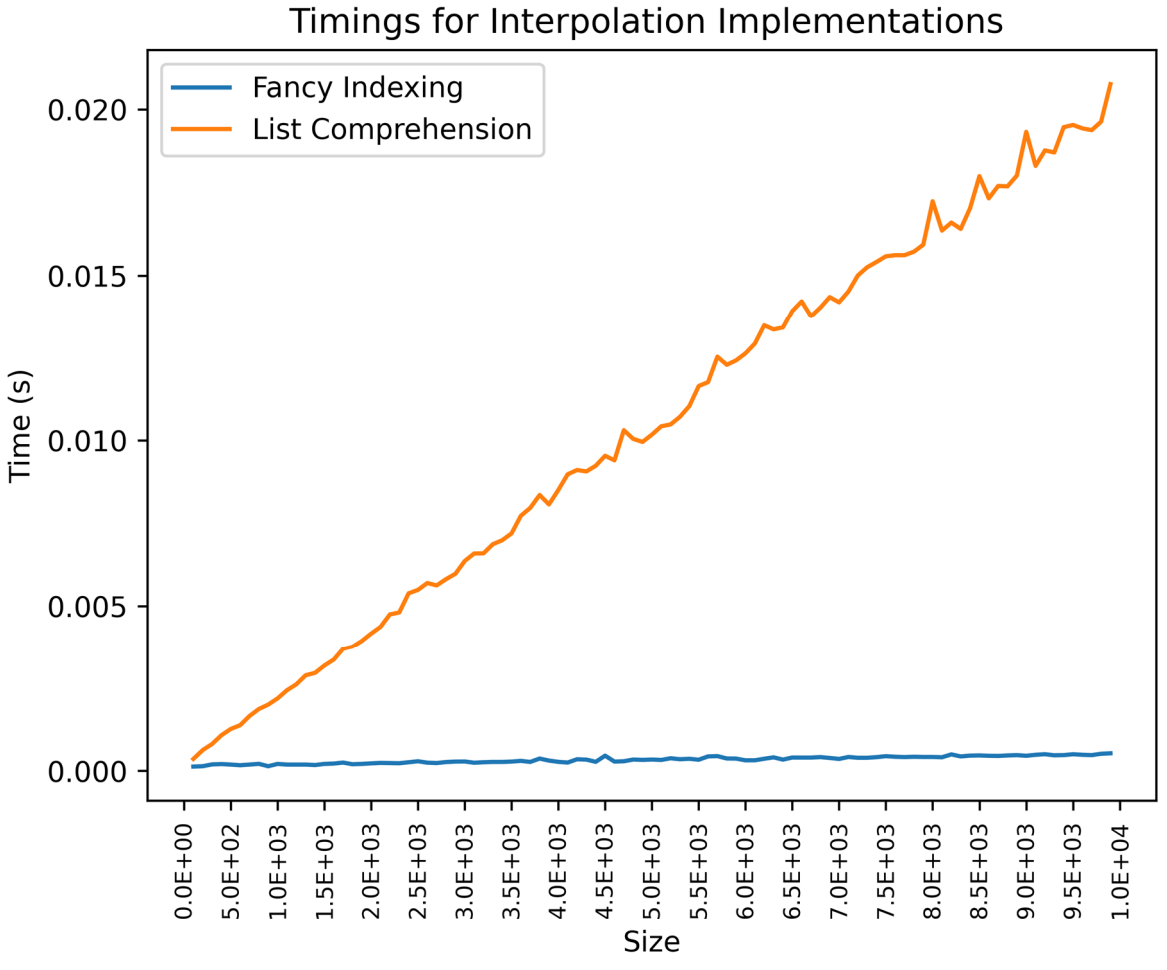
Fancy Indexing

```
v = y[k] + s * (d[k] + s * (c[k] + s * b[k]))
```

Interpolation

Discussion: Interpolation

	Slope
List Comprehension	$2.0237 \cdot 10^{-6}$
Fancy Indexing	$3.4812 \cdot 10^{-8}$



Non-Linear Equation Solver

Packages

- Bisection
- Newton's Method
- Secant Method
- Inverse Quadratic Interpolation
- Brent-Dekker's Method

Non-Linear Equation Solver

Discussion

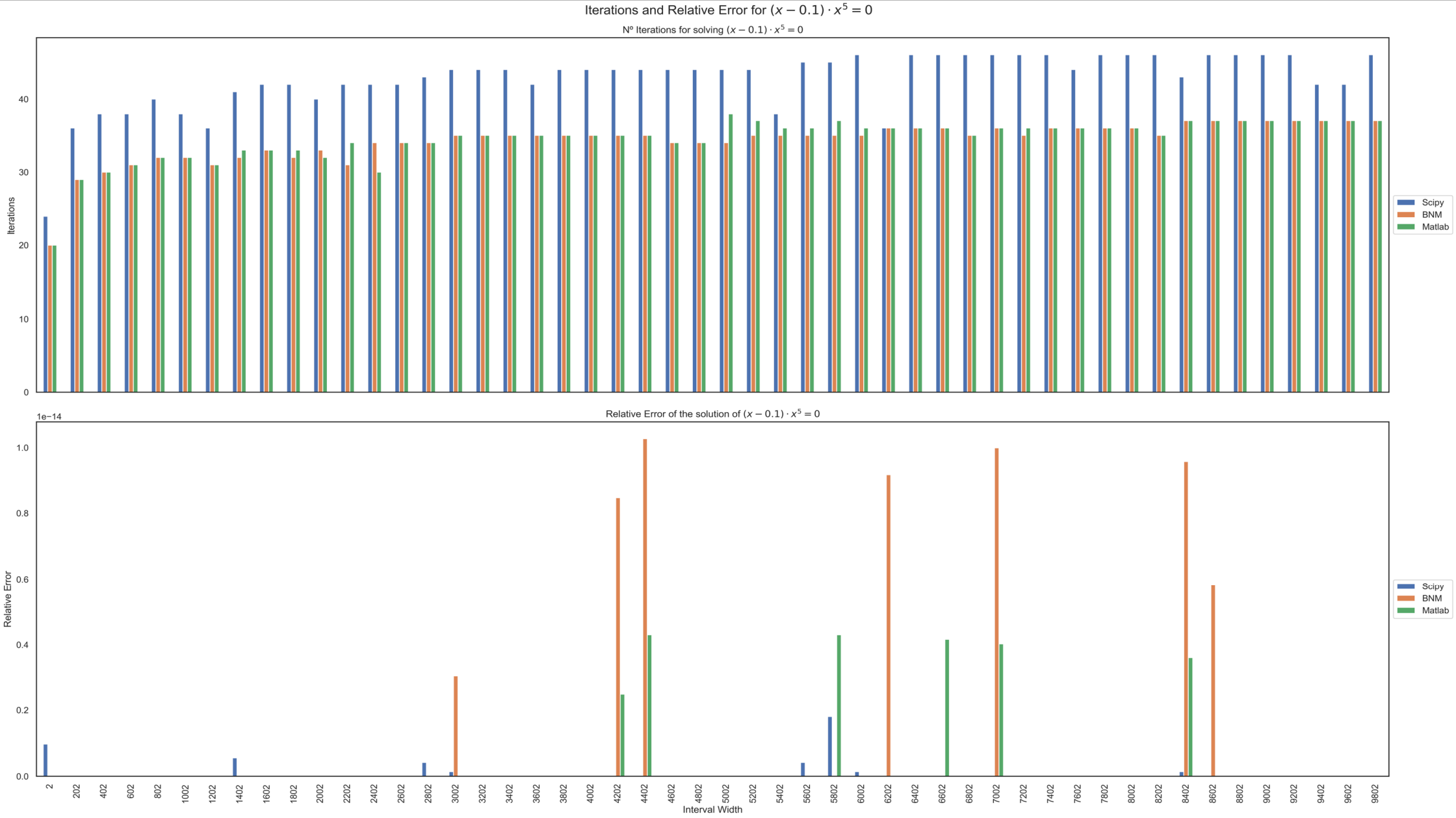
Original Brent-Dekker

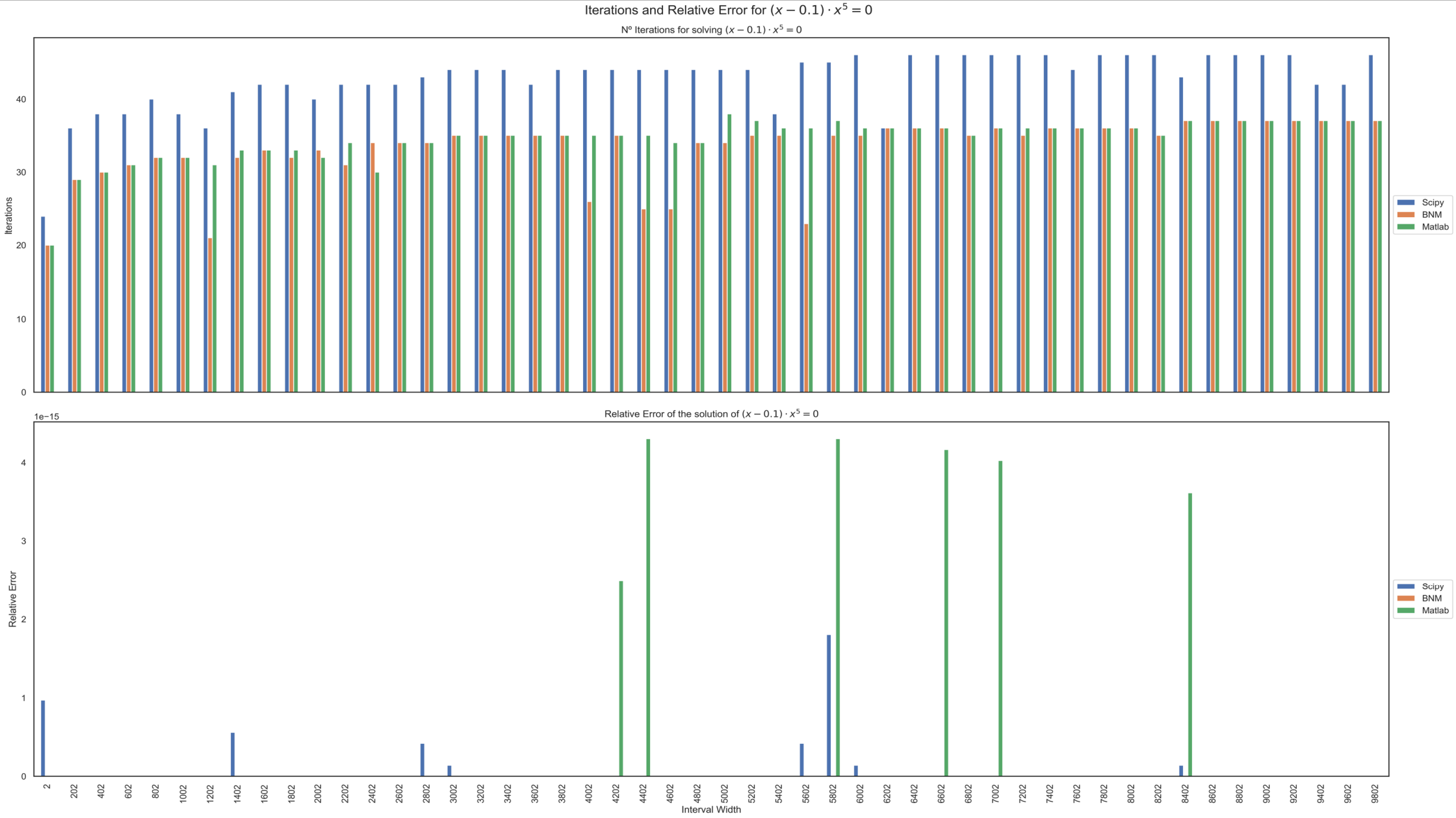
Vs.

MATLAB's Brent-Dekker

Vs.

Scipy's Brent-Dekker


Iterations and Relative Error for $(x - 0.1) \cdot x^5 = 0$
Nº Iterations for solving $(x - 0.1) \cdot x^5 = 0$
Iterations
0
10
20
30
40
Scipy
BNM
Matlab
Relative Error of the solution of $(x - 0.1) \cdot x^5 = 0$
1e−14
Relative Error
0.0
0.2
0.4
0.6
0.8
1.0
Scipy
BNM
Matlab
2
202
402
602
802
1002
1202
1402
1602
1802
2002
2202
2402
2602
2802
3002
3202
3402
3602
3802
4002
4202
4402
4602
4802
5002
5202
5402
5602
5802
6002
6202
6402
6602
6802
7002
7202
7402
7602
7802
8002
8202
8402
8602
8802
9002
9202
9402
9602
9802
Interval Width

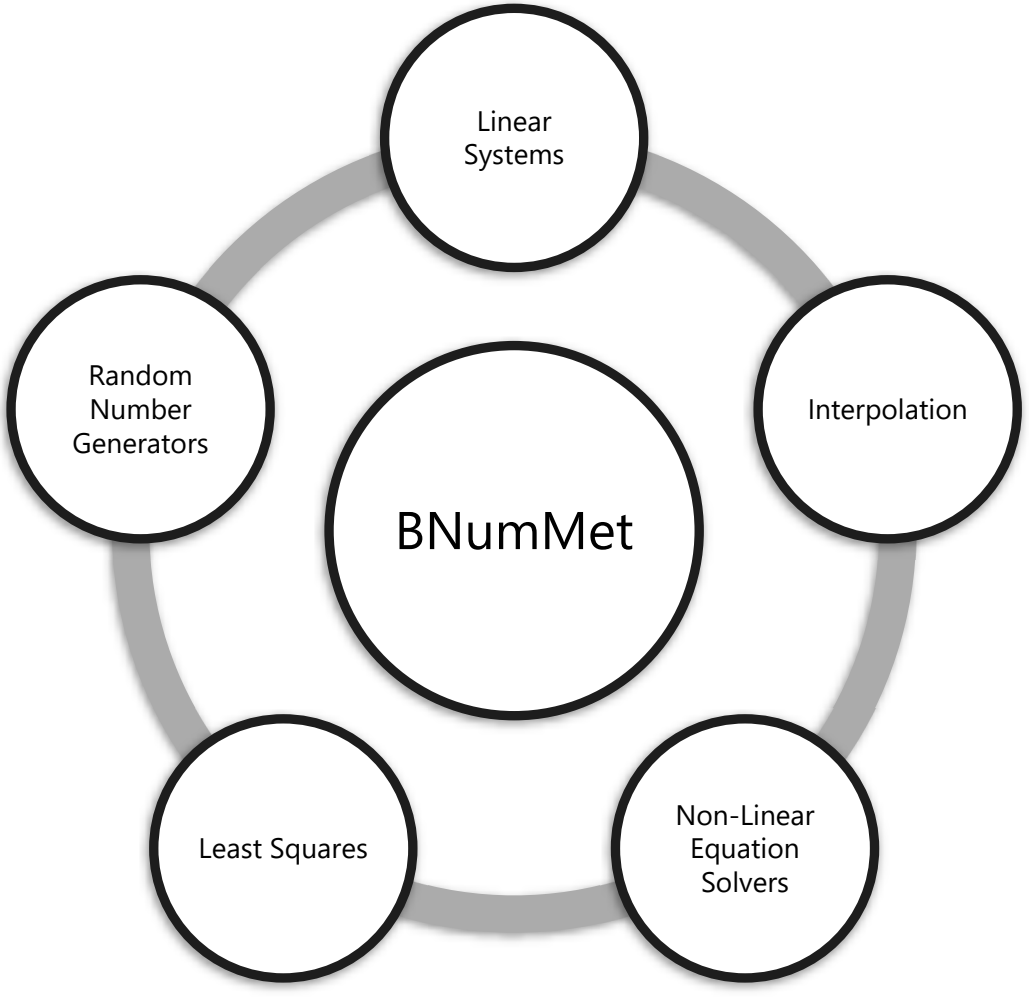

Iterations and Relative Error for $(x - 0.1) \cdot x^5 = 0$
Nº Iterations for solving $(x - 0.1) \cdot x^5 = 0$
Iterations
0
10
20
30
40
Scipy
BNM
Matlab
Relative Error of the solution of $(x - 0.1) \cdot x^5 = 0$
1e−15
Relative Error
0
1
2
3
4
Scipy
BNM
Matlab
2
202
402
602
802
1002
1202
1402
1602
1802
2002
2202
2402
2602
2802
3002
3202
3402
3602
3802
4002
4202
4402
4602
4802
5002
5202
5402
5602
5802
6002
6202
6402
6602
6802
7002
7202
7402
7602
7802
8002
8202
8402
8602
8802
9002
9202
9402
9602
9802
Interval Width

Random Number Generators

Packages

Lehmer's	Linear Congruential Algorithm
	$x_{n+1} = (a \cdot x_n + c)\, mod(n)$
Marsaglia's	Subtract With Borrow or Add with Borrow
	$f(x_1, x_2, \dots, x_r, c) = \begin{cases} (x_2, \dots, x_r, x_{r+1-s} - x_1 - c, 0) & x_{r+1-s} - x_1 - c \geq 0 \\ (x_2, \dots, x_r, x_{r+1-s} - x_2 - c + b, 1) & x_{r+1-s} - x_1 - c < 0 \end{cases}$
Mersenne-Twister	Up to date RNG
	Used in Python & R

Ending


Linear Systems
Random Number Generators
BNumMet
Interpolation
Least Squares
Non-Linear Equation Solvers

Ending

LIVE DEMO

Summary

BNumMet	Scholarly Numerical Methods	*Go-To guide for students*
	Visual Interface	*Capture the Essence of the Methods*
	Introduction to Python, Numerical Methods & Jupyter	*Original didactic tool for undergraduates & teachers*
	Efficient, Original & Competitive	*Compared and Analysed*
	Open-Sourced	*"Learning is priceless"*

Thank you for your attention

Any questions?

NAME	Fernando Bellido Pazos
MASTER	Applied & Computational Mathematics
TUTOR	Juan Manuel Perez Pardo