

Scientific Computing With Rust

Srinath Kailasa

Department of Mathematics
University College London

April 18, 2022



Table of Contents

Overview

Expressing Science with Software

Current Research Directions

Concluding Remarks

Table of Contents

Overview

Expressing Science with Software

Current Research Directions

Concluding Remarks

Team and Research Focus

Research Focus:

1. Numerical Analysis & Scientific Computing
2. PDEs: Acoustics, Electromagnetics, Electrostatics
3. High-Performance Computing and Software Engineering



Prof. Timo Betcke
@BetckeTimo



Srinath Kailasa
@SrinathKailasa



Ignacia
Fierro-Piccardo
@ignaciafpicc

My Research

'Science with Computers and Maths'

1. High Performance Computing
2. Heterogenous Computing
3. Software Engineering
4. Problems in Physics and Engineering

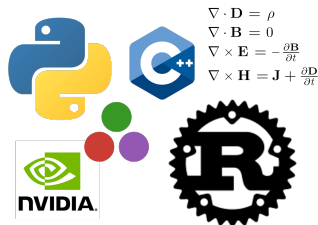


Table of Contents

Overview

Expressing Science with Software

Current Research Directions

Concluding Remarks

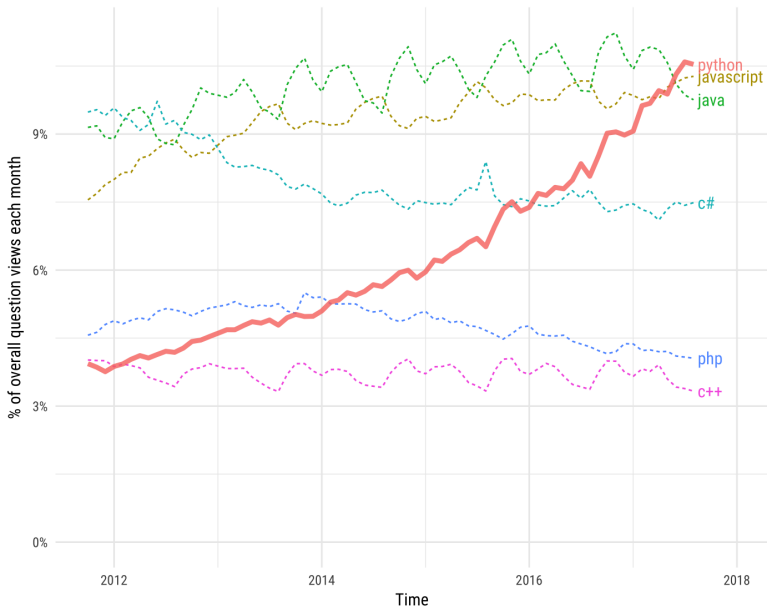
Expressing Scientific Problems With Software

There is no 'best' language for expressing scientific problems with software.

Though Python has emerged as a defacto standard amongst scientists and engineers for a broad spectrum of problems.

Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



The Two Language Problem

1. Languages suited for human needs, are less efficient for computers to run.
2. Languages easy for computers to run efficiently, are correspondingly less easy for humans to use!

Why Rust?

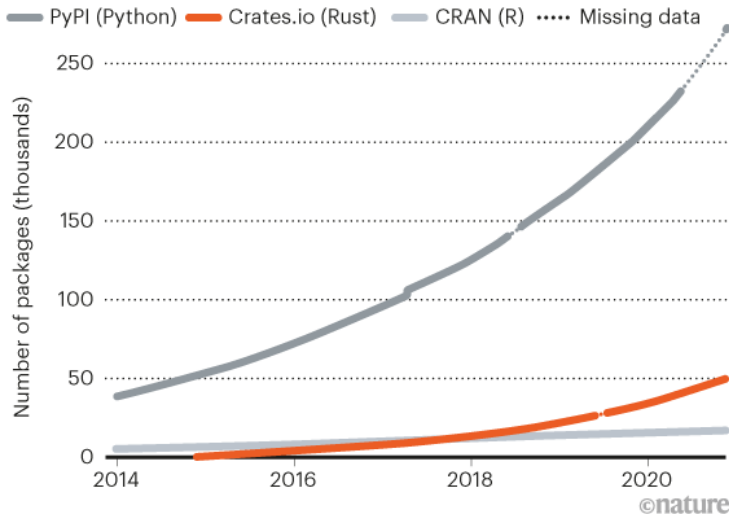
Don't many of the 'two language' problems still exist?

Pros of Rust:

1. Cargo is awesome!
2. Rust is Fast
3. Foreign language interfaces are easy
4. Easy to learn (harder to master)
5. Traits
6. Borrow Checker

RUST RISING

The Rust packages repository crates.io has grown sharply since 2016, mirroring the rapid uptake of the language.



<https://www.nature.com/articles/d41586-020-03382-2>

State of Scientific Computing in Rust

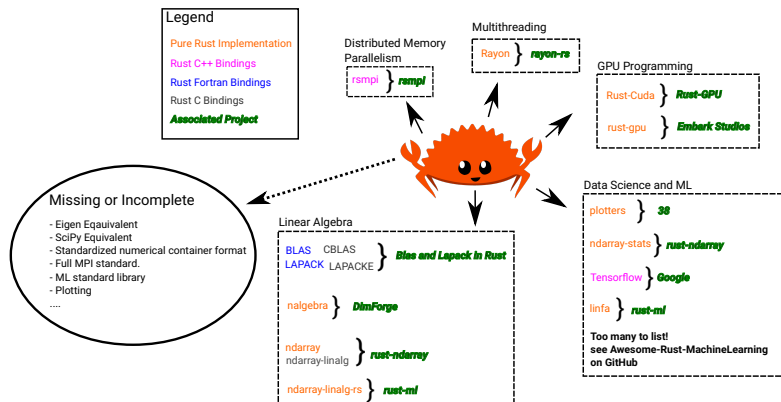


Table of Contents

Overview

Expressing Science with Software

Current Research Directions

Concluding Remarks

A Splash of Differential Equations

What is an Ordinary Differential Equation?

Imposition of a relationship between functions of a independent single variable and its derivatives.

e.g. Newton's Second Law of Motion (in 1 Dimension)

$$m \frac{d^2 x}{dt^2} = F(x(t)) \quad (1)$$

A Splash of Differential Equations

What is a Partial Differential Equation?

Imposition of a relationship between functions of a multiple independent variables and their derivatives.

e.g. The Heat Equation (in 3D Cartesian Coordinates,

$$\Delta = \frac{d^2}{dx^2} + \frac{d^2}{dy^2} + \frac{d^2}{dz^2})$$

$$\frac{du}{dt} = \Delta u \quad (2)$$

Where $u(x, y, z, t)$.

Variational Methods for PDEs I

Finite Element Method (FEM):

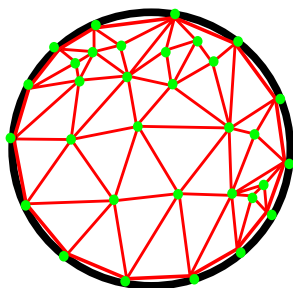
1. Meshes over volume.
2. Results in sparse matrices.
3. Requires explicit boundary conditions.

Boundary Element Method (BEM):

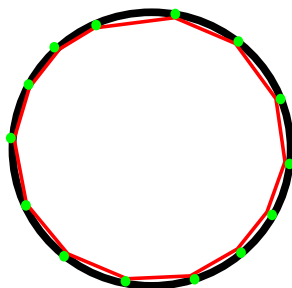
1. Meshes over surface.
2. Results in full matrices.
3. Boundary conditions captured in formulation.

Variational Methods for PDEs II

Consider 2D problem:



FEM



BEM

Laplace Interior Boundary Value Problem I

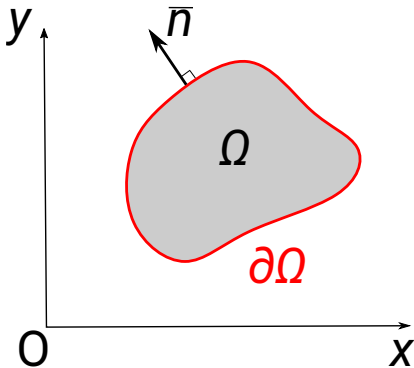
Want to solve

$$\frac{\partial \phi}{\partial x^2} + \frac{\partial \phi}{\partial y^2} = 0 \quad (3)$$

Inside a region of interest Ω
which has some boundary $\partial\Omega$.

We have conditions for the
solution on the boundary:

1. $\frac{\partial \phi(\partial\Omega)}{\partial n}$ - Neumann type.
2. $\phi(\partial\Omega)$ - Dirichlet type



Laplace Interior Boundary Value Problem II

Find a solution in terms of a *boundary integral*,

$$\lambda(\xi, \eta)\phi(\xi, \eta) = \int_{\partial\Omega} [\phi(x, y) \frac{\partial}{\partial n}(\Phi(x, y; \xi, \eta)) - \Phi(x, y; \xi, \eta) \frac{\partial}{\partial n}(\phi(x, y))] ds(x, y) \quad (4)$$

Where,

$$\lambda(\xi, \eta) = \begin{cases} 0 & (\xi, \eta) \notin \Omega \cup \partial\Omega \\ 1/2 & (\xi, \eta) \in \partial\Omega \\ 1 & (\xi, \eta) \in \Omega \end{cases} \quad (5)$$

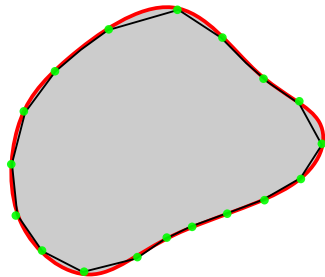
and the *Fundamental Solution*,

$$\Phi(x, y; \xi, \eta) = \frac{1}{4\pi} \ln[(x - \xi)^2 + (y - \eta)^2] \quad (6)$$

Laplace Interior Boundary Value Problem III

$$\lambda(\xi, \eta)\phi(\xi, \eta) = \int_{\partial\Omega} [\phi(x, y) \frac{\partial}{\partial n}(\Phi(x, y; \xi, \eta)) - \Phi(x, y; \xi, \eta) \frac{\partial}{\partial n}(\phi(x, y))] ds(x, y) \quad (7)$$

Laplace Interior Boundary Value Problem IV



Laplace Interior Boundary Value Problem V

Approximate with N constant elements, with a midpoint value.

$$\begin{aligned} \lambda(\xi, \eta)\phi(\xi, \eta) = & \sum_{k=1}^N \overline{\phi(x, y)}^{(k)} \int_{\partial\Omega} \frac{\partial}{\partial n}(\Phi(x, y; \xi, \eta)) ds(x, y) \\ & - \overline{\frac{\partial}{\partial n}(\phi(x, y))}^{(k)} \int_{\partial\Omega} \Phi(x, y; \xi, \eta) ds(x, y) \end{aligned} \quad (8)$$

Laplace Interior Boundary Value Problem VI

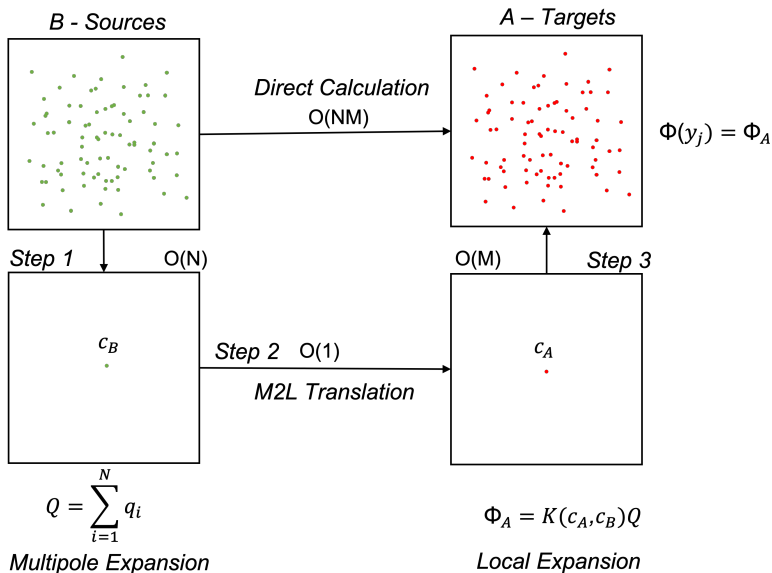
Final Matrix Vector product (matvec) relating points on boundary, to solution in interior of the form,

$$Ax = b \quad (9)$$

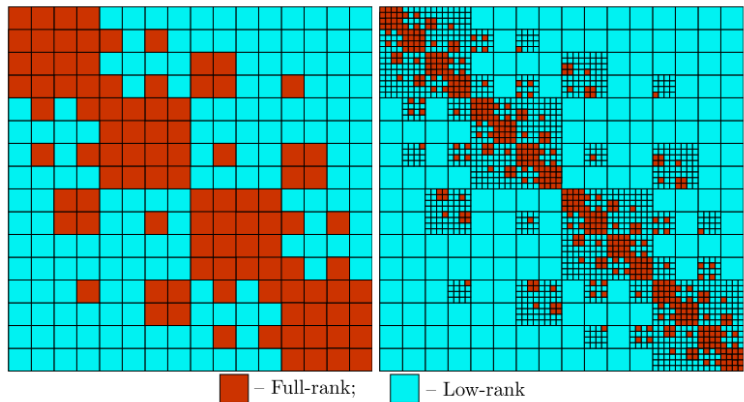
Where A is the coefficient matrix, x is the boundary data and b is the solution.

1. $O(N^2)$ scaling with N elements.
2. Each matrix element also involves calculating horrible maths.

Fast Multipole Methods

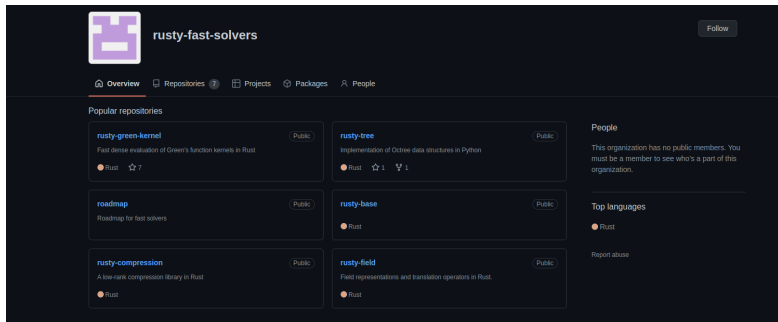


Fast Direct Solvers



Ambisekaran, S & Darve, E. *The Inverse Fast Multipole Method*, arXiv:1407.1572v1 (2014)

Rusty Fast Solvers Project I



The screenshot shows the GitHub organization page for 'rusty-fast-solvers'. At the top, there is a purple logo and the organization name. Below this is a navigation bar with links to Overview, Repositories, Projects, Packages, and People. The main content area is titled 'Popular repositories' and displays a grid of six repository cards. Each card includes the repository name, a brief description, a 'Public' badge, and a Rust logo. The repositories are: 'rusty-green-kernel' (Fast dense evaluation of Green's function kernels in Rust, 7 stars), 'rusty-tree' (Implementation of Octree data structures in Python, 1 star, 1 fork), 'roadmap' (Roadmap for fast solvers), 'rusty-base' (Rust), 'rusty-compression' (A low-rank compression library in Rust), and 'rusty-field' (Field representations and translation operators in Rust). To the right of the repository grid, there is a 'People' section stating that the organization has no public members, and a 'Top languages' section showing Rust as the primary language. A 'Report abuse' link is also present.

rusty-fast-solvers Follow

Overview Repositories Projects Packages People

Popular repositories

- rusty-green-kernel** (Public)
Fast dense evaluation of Green's function kernels in Rust
Rust 7
- rusty-tree** (Public)
Implementation of Octree data structures in Python
Rust 1 1
- roadmap** (Public)
Roadmap for fast solvers
- rusty-base** (Public)
Rust
- rusty-compression** (Public)
A low-rank compression library in Rust
Rust
- rusty-field** (Public)
Field representations and translation operators in Rust.
Rust

People
This organization has no public members. You must be a member to see who's a part of this organization.

Top languages
Rust

Report abuse

<https://github.com/rusty-fast-solvers>

Rusty Fast Solvers Project II

Checklist:

1. **Rusty Green Kernel** - Optimized math kernel operations.
2. **Rusty Tree** - Distributed Octrees.
3. **Hyksort** - Sorting algorithm for distributed arrays.
4. **Rusty Compression** - Randomized compression library.
5. **Rusty Field** - Field representations.
6. **Rusty FMM** - Fast Multipole Method.
7. **Rusty Inverse** - Fast Direct Solvers.

Rusty Fast Solvers Project III

Rusty Tree is the most complete work item, with full Python bindings.

It's a distributed octree implementation, parallelized with MPI.

<https://github.com/rusty-fast-solvers/rusty-tree>

Maturin

Maturin is a tool for developing Python bindings for Rust using its foreign function interface.

Let's look at a demo project together:

<https://github.com/skailasa/pyrustmpi>

Table of Contents

Overview

Expressing Science with Software

Current Research Directions

Concluding Remarks

Conclusion

1. We're building Rust infrastructure for the $O(N)$ forward and inverse application of a large class of integral operators that incorporates multiple levels of parallelism.
2. Rust makes building and distributing scientific software easy.
3. Ecosystem of numerics tools is growing.
4. We're excited to be part of a larger movement!