Scientific Computing With Rust

Srinath Kailasa

Department of Mathematics
University College London

October 10, 2022

Table of Contents

Table of Contents

Team and Research Focus

Research Focus:

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

Srinath Kailasa
@SrinathKailasa

https://github.com/skailasa https://github.com/betckegroup
https://github.com/rusty-fast-solvers

My Research

'Science with Computers and Maths'

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

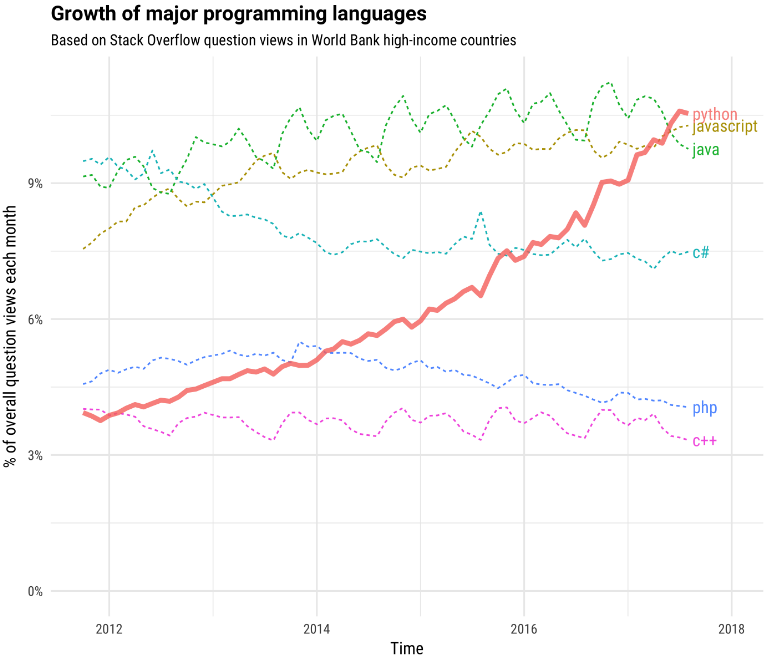
$$
\begin{aligned}
\nabla \cdot \mathbf{D} &= \rho \\
\nabla \cdot \mathbf{B} &= 0 \\
\nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}
\end{aligned}
$$

Table of Contents

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.



https://stackoverflow.blog/2017/09/06/incredible-growth-python/

The Two Language Problem

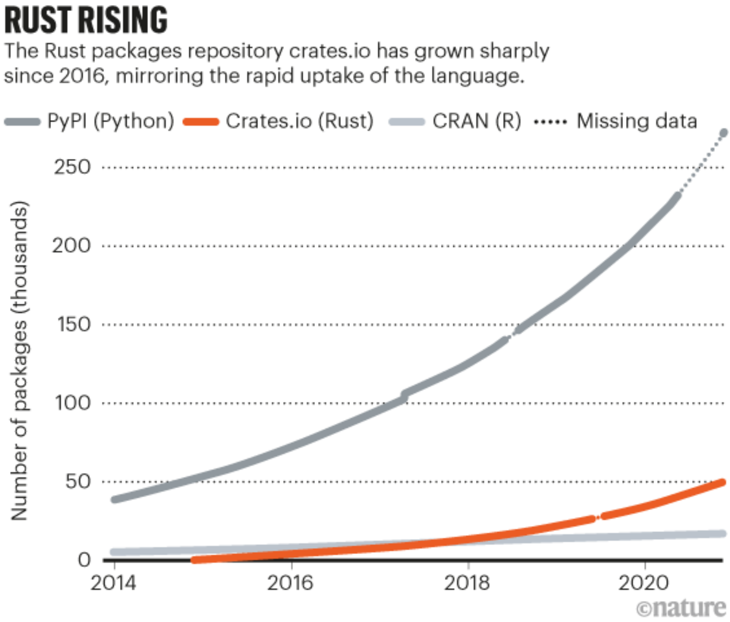
1. Languages suited for human needs, are less efficient for computers to run.
2. Languages easy for computers to run efficiently, are correspondingly less easy for humans to use!

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



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

State of Scientific Computing in Rust

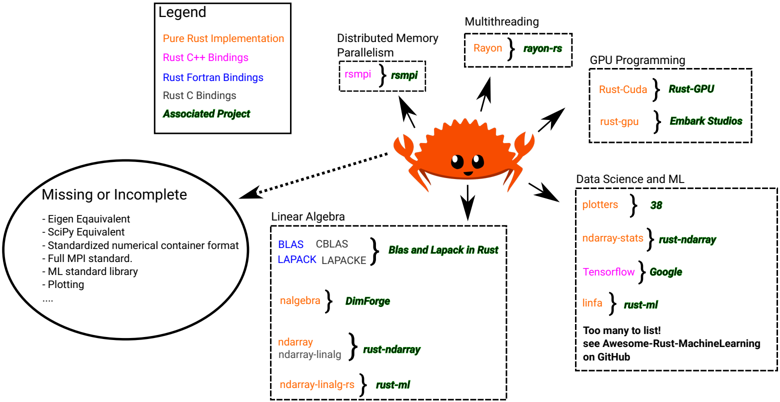


Table of Contents

A Splash of Differential Equations

We are interested in numerically solving Partial Differential Equations

e.g. The Heat Equation (in 3D Cartesian Coordinates, $\Delta = \frac{d^2}{dx^2} + \frac{d^2}{dy^2} + \frac{d^2}{dz^2}$)

$$(\Delta + k^2)u(x) = f \quad x \in \Omega \tag{1}$$

$$u(x) = g(x), \quad x \in \partial\Omega \tag{2}$$

Where $\Omega \subseteq \mathbb{R}^d$, and $d = 2$ or $d = 3$.

Variational Methods for PDEs I

Finite Element Method (FEM):

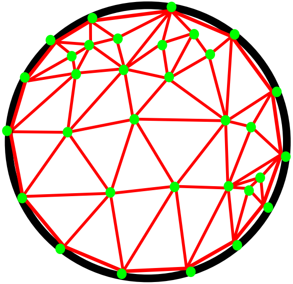
1. Meshes over volume.
2. Results in sparse matrices.
3. Requires explicit boundary conditions.

Boundary Element Method (BEM):

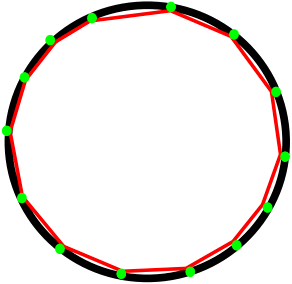
1. Meshes over surface.
2. Results in full matrices.
3. Boundary conditions captured in formulation.

Variational Methods for PDEs II

Consider 2D problem:



FEM



BEM

Laplace Interior Boundary Value Problem I

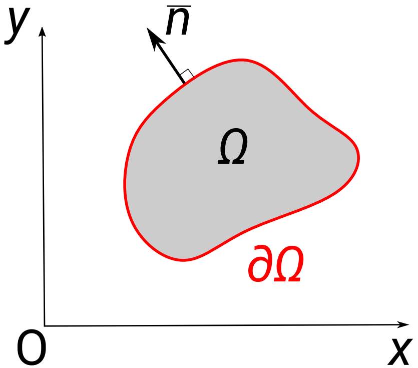
Want to solve

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

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*,

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

Where,

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

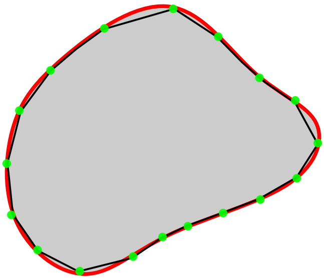
and the *Fundamental Solution*,

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

Laplace Interior Boundary Value Problem III

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

Laplace Interior Boundary Value Problem IV



Laplace Interior Boundary Value Problem V

Approximate with N constant elements, with a midpoint value.

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

Laplace Interior Boundary Value Problem VI

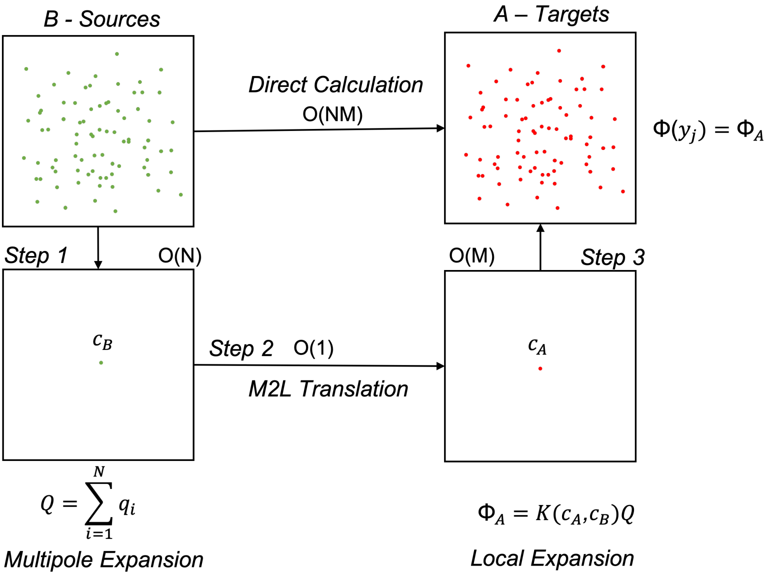
Final Matrix Vector product (matvec) relating points on boundary, to solution in interior of the form,

$$Ax = b \tag{9}$$

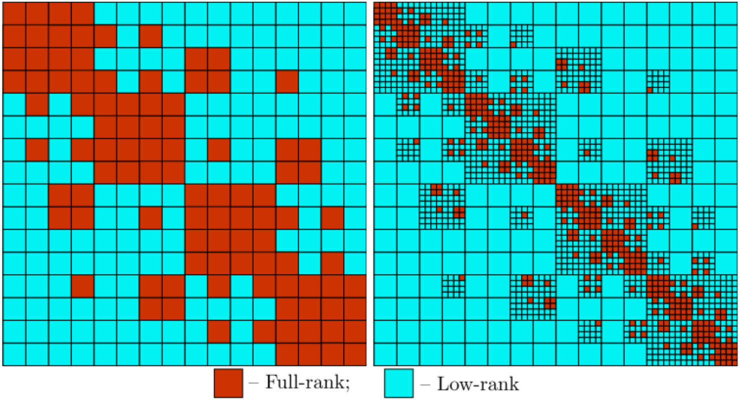
Where A is the coefficient matrix, x is the boundary data and b is the solution.

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

Fast Multipole Methods

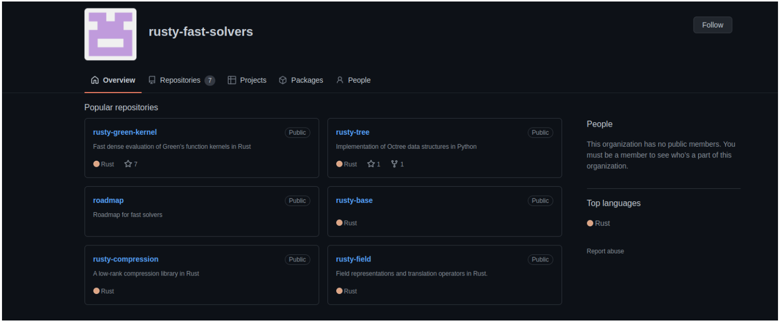


Fast Direct Solvers



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

Rusty Fast Solvers Project I



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

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!