IBLGF

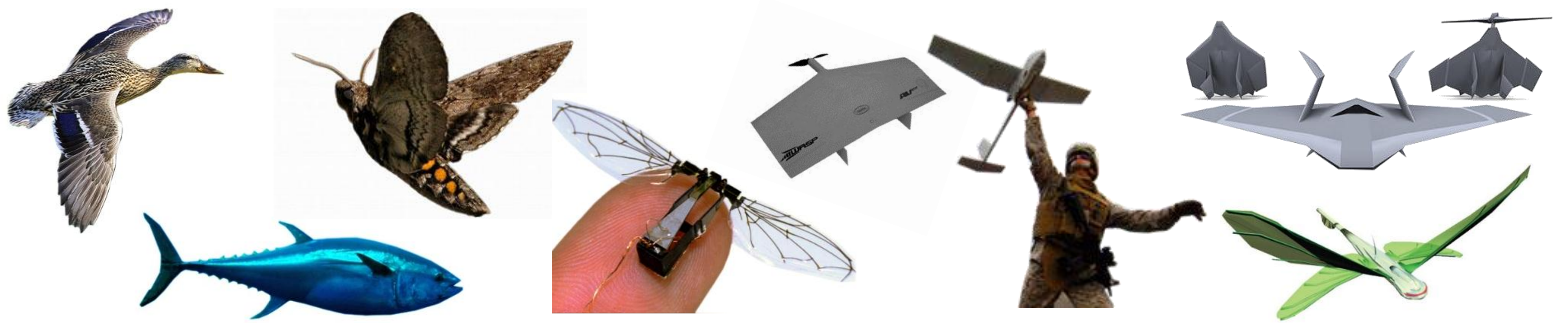
An Immersed Boundary Method for the Incompressible Navier-Stokes Equations based on the Lattice Green's Function Method

Sebastian Liska

California Institute of Technology

ACM114 Project Presentation
April 25th, 2011

Motivation



Modern applications require a better understanding of flow past complicated geometries at low Reynolds numbers

:: three-dimensional :: unsteady ::
:: separated/detached :: wide range of scales ::
:: external flows :: time-dependent geometries ::

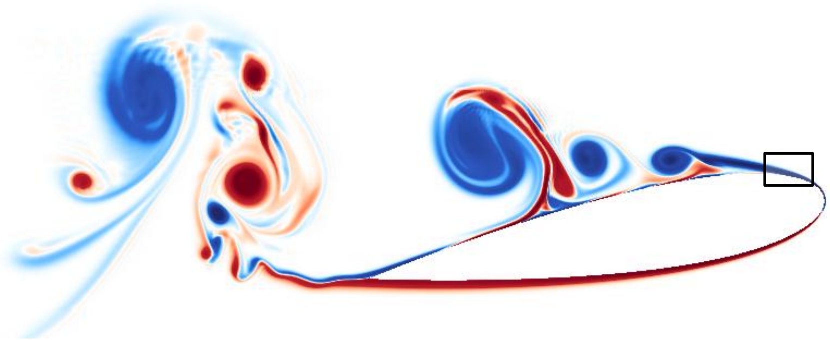
Motivation

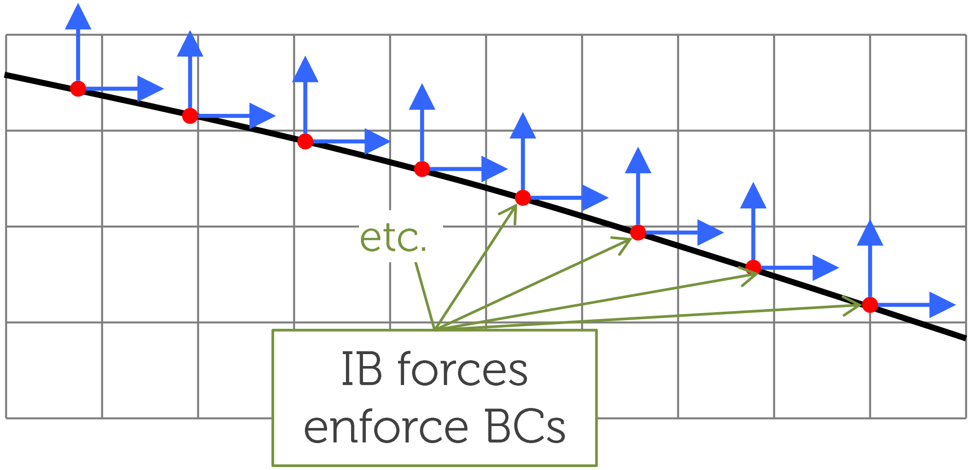
- Explore flow physics through direct numerical simulation
- Immersed Boundary Lattice Green's Function (IBLGF) method
  - Incompressible Navier-Stokes
  - Discretization based on an infinite Cartesian grid (but operations done a on small finite portion)
  - Immersed surfaces are included by Immersed Boundary (IB) method

:: 3D :: parallel :: robust :: flexible :: fast/efficient ::

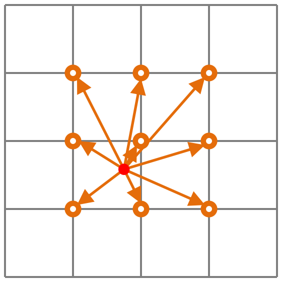
Immersed Boundary Method

- Flow solved on an Eulerian grid
- IB generated by forces at Lagrangian points





- Communication between grids via discrete Delta functions



Fluid with IB Equations

Constant density Navier-Stokes equations with immersed boundary

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} + \overbrace{\int_s \mathbf{f}\left(\xi(s,t)\right) \delta\left(\xi - \mathbf{x}\right)\, ds}^{\text{immersed boundary forcing}}$$

$$\nabla \cdot \mathbf{u} = 0$$

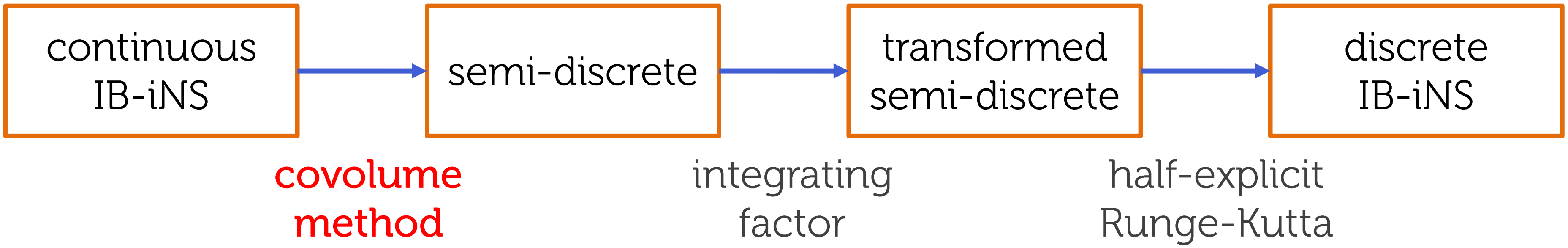
$$\mathbf{u}\left(\xi(s,t)\right) = \underbrace{\int_{\mathbf{x}} \mathbf{u}(\mathbf{x}) \delta\left(\xi - \mathbf{x}\right)\, d\mathbf{x} = \mathbf{u}_B\left(\xi(s,t)\right)}_{\text{no-slip at immersed boundary points}}$$

Discretization

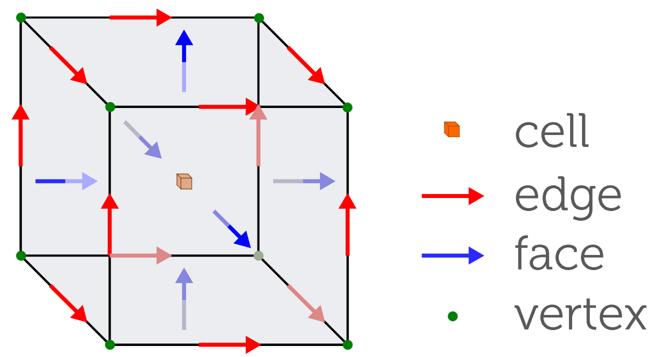
continuous IB-iNS $\xrightarrow{\text{covolume method}}$ semi-discrete $\xrightarrow{\text{integrating factor}}$ transformed semi-discrete $\xrightarrow{\text{half-explicit Runge-Kutta}}$ discrete IB-iNS

Discretization based on an infinite Cartesian grid with boundary conditions applied at infinity

Discretization



Discrete operators have similar properties to continuous analogs



R. A. Nicolaides, X. Wu, SIAM J. Numer. Anal. 34 (1997)

Discretization

continuous IB-iNS → semi-discrete → transformed semi-discrete → discrete IB-iNS

covolume method

integrating factor

half-explicit Runge-Kutta

Eliminate viscous terms using an integrating factor technique

$$\frac{du}{dt} = Lu,\ u(t=0) = u_0$$

$$u = u_0 e^{tL}$$

Discretization

continuous IB-iNS → semi-discrete → transformed semi-discrete → discrete IB-iNS

covolume method | integrating factor | **half-explicit Runge-Kutta**

½ERK time integration for convective terms and algebraic constraints (div-free & no-slip)

$$\frac{dv}{dt} = \mathcal{F}(v, z, t)$$
$$0 = \mathcal{G}(u, z, t)$$

Nr.	tree	order	order condition
1		1	$\sum b_i = 1$
2		2	$\sum b_i c_i = \frac{1}{2}$
3		3	$\sum b_i c_i^2 = \frac{1}{3}$
4		3	$\sum b_i a_{ij} c_j = \frac{1}{6}$
5		3	$\sum b_i c_i \omega_{ij} c_{j+1}^2 = \frac{2}{3}$
6		3	$\sum b_i \omega_{ij} c_{j+1}^2 \omega_{ik} c_{k+1}^2 = \frac{4}{3}$

V. Brasey, E. Hairer, SIAM J. Numer. Anal. 2 (1993)

Discretization

continuous IB-iNS $\xrightarrow{\text{covolume method}}$ semi-discrete $\xrightarrow{\text{integrating factor}}$ transformed semi-discrete $\xrightarrow{\text{half-explicit Runge-Kutta}}$ discrete IB-iNS

… details …

Most expensive operations involve solving the Poisson equations used to enforce the div-free and no-slip constraints

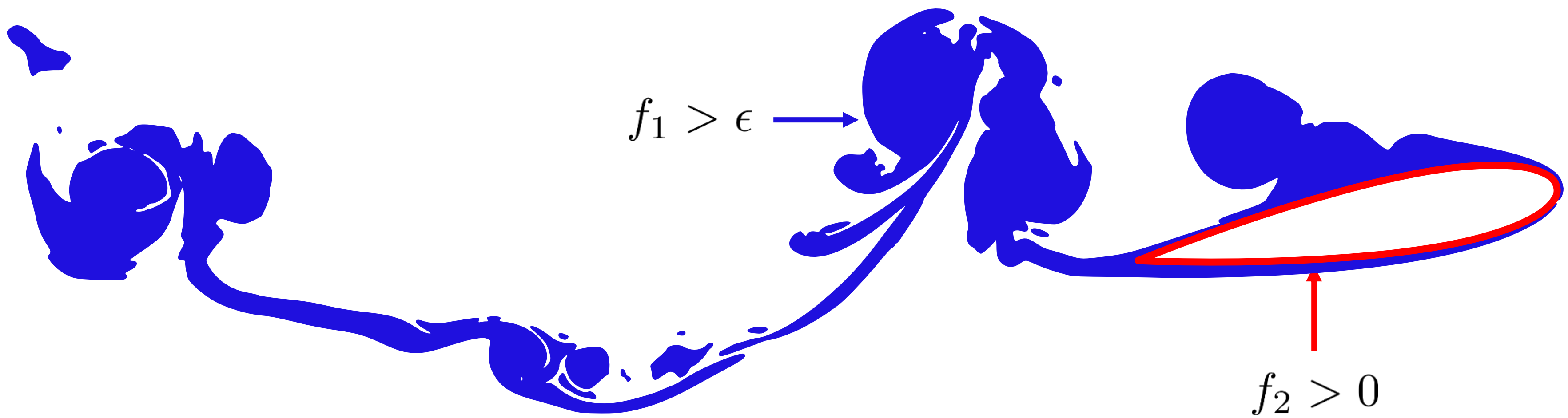
$$Lu = f$$

$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$

Poisson Problems

$$Lu = f_q$$

$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$

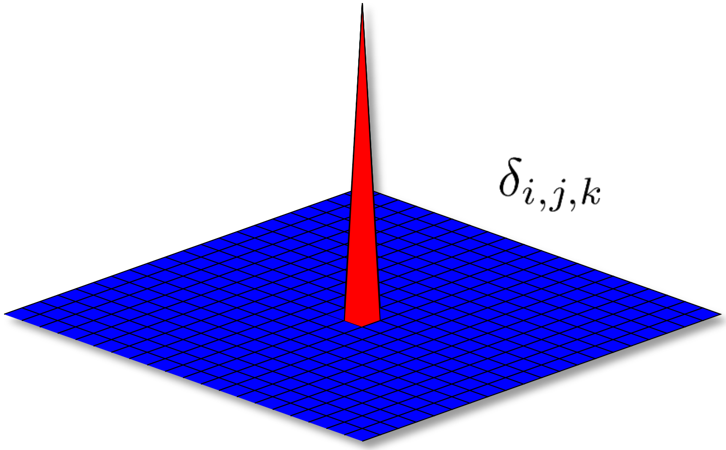


Lattice Green's Function

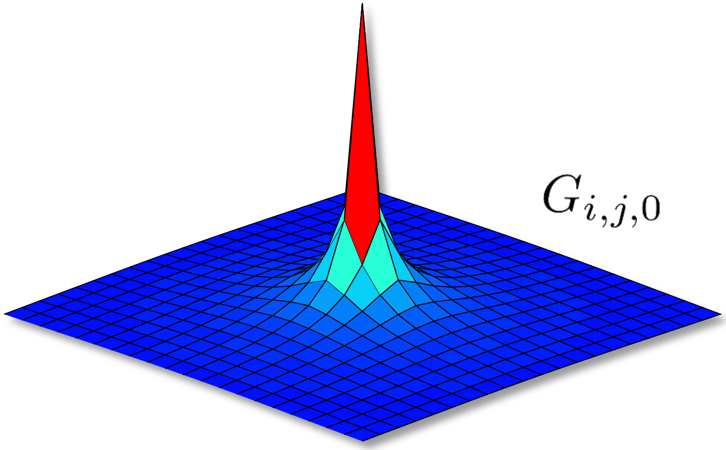
Discrete analog of the Green's function for Laplace's equation

$$LG = \delta_{i,j,k}$$

$$G_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$



$\delta_{i,j,k}$



$G_{i,j,0}$

W. H. Mccrea & F. J. W. Whipple ('40); R. J. Duffin ('58); O. Buneman ('71); P.G. Martinsson & G.J. Rodin ('02)

Poisson Problems

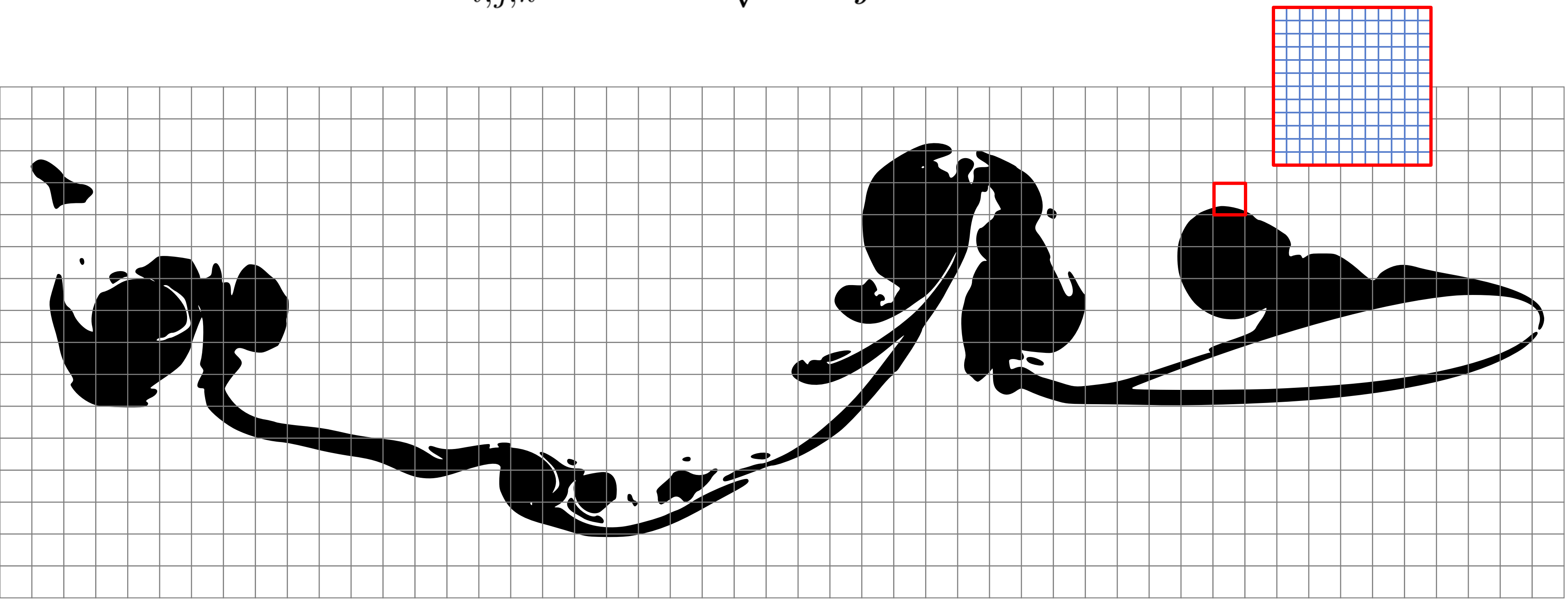
$$Lu = f_q$$

$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$

Poisson Problems

$$Lu = f_q$$

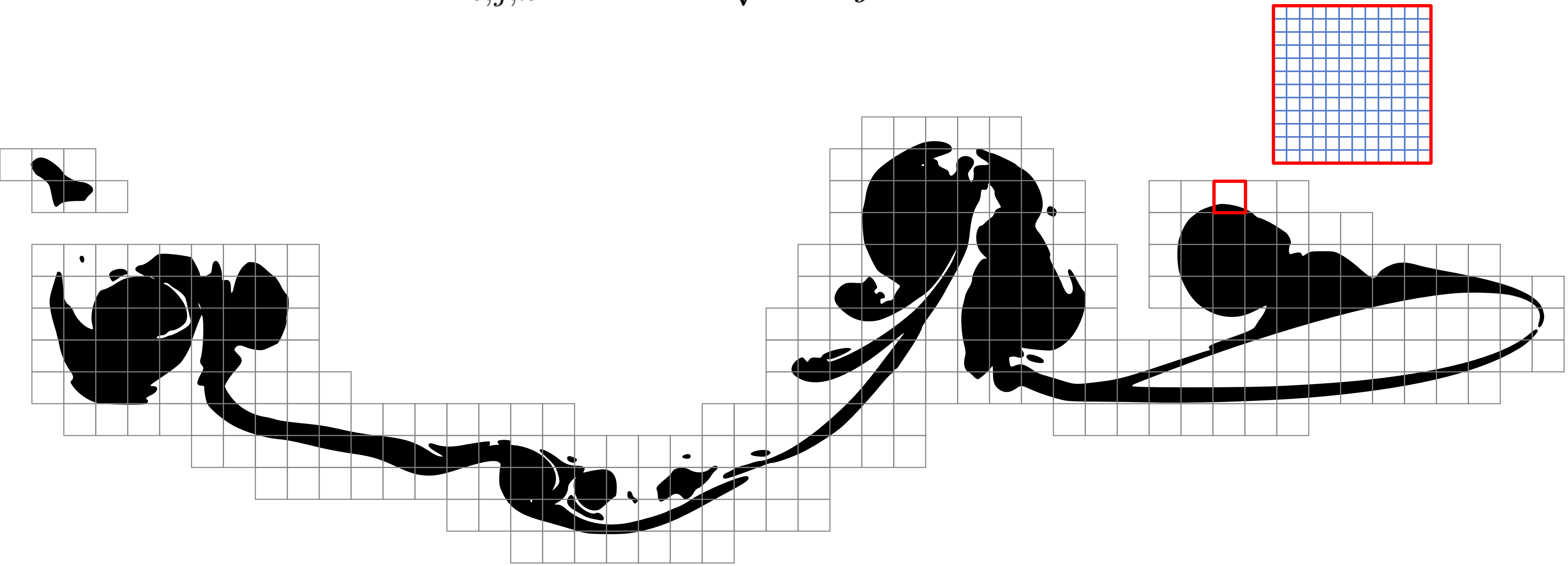
$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$



Poisson Problems

$$Lu = f_q$$

$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$

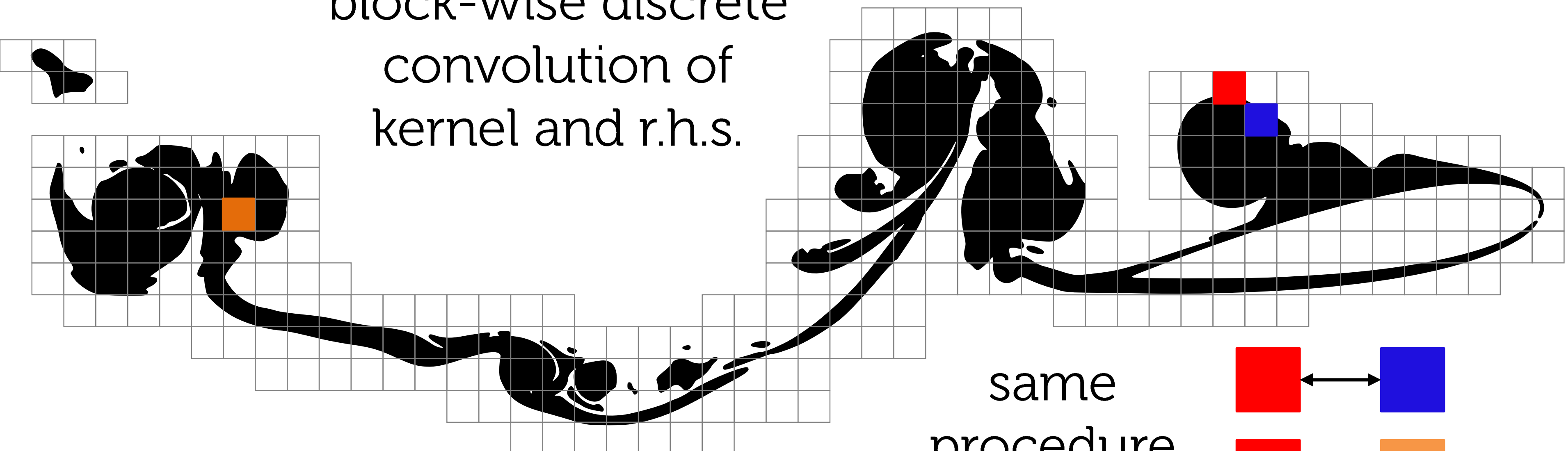


Poisson Problems: Present Solver

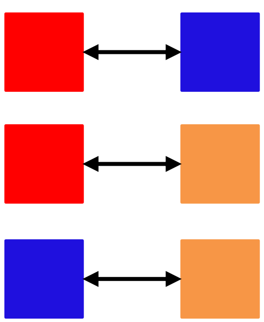
$$Lu = f_q$$

$$u_{i,j,k} \longrightarrow 0 \text{ as } \sqrt{i^2 + j^2 + k^2} \longrightarrow \infty$$

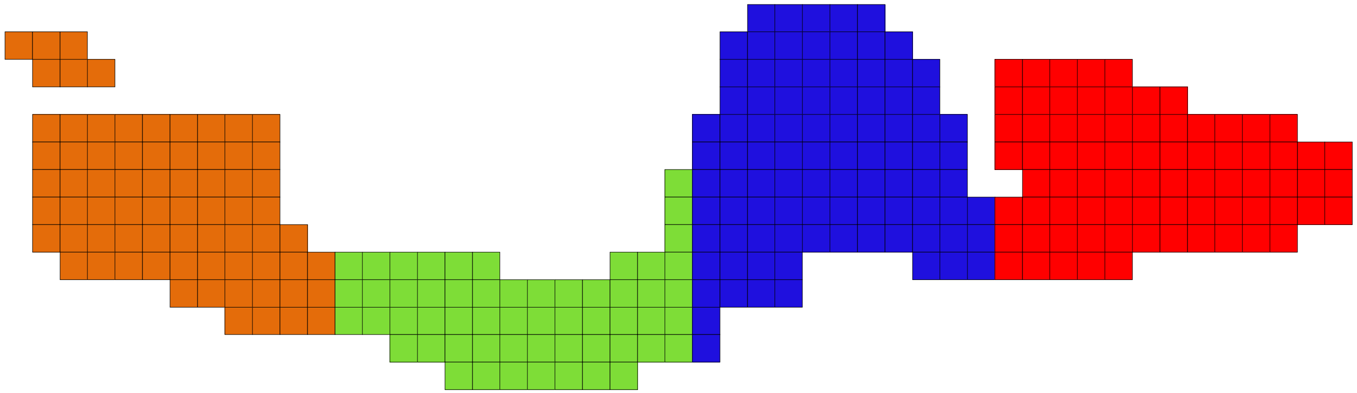
block-wise discrete convolution of kernel and r.h.s.



same procedure & cost $\Longrightarrow \mathcal{O}(n^2)$



Poisson Problems: Present Solver



for P **in** Processors

  for B **in** P.Blocks

    C[**id**(B)] ← 0

    for MyB **in** MyProcessor.Blocks

      C[**id**(B)] ← C[**id**(B)] + **Interaction**(from = MyB, to = B)

get and package contributions from my blocks to target blocks

  if **id**(MyProcessor) == **id**(P)

    Pck ← **sum** C from all Processors

global reduction

    for B **in** P.Blocks

      B.Result ← Pck[**id**(B)]

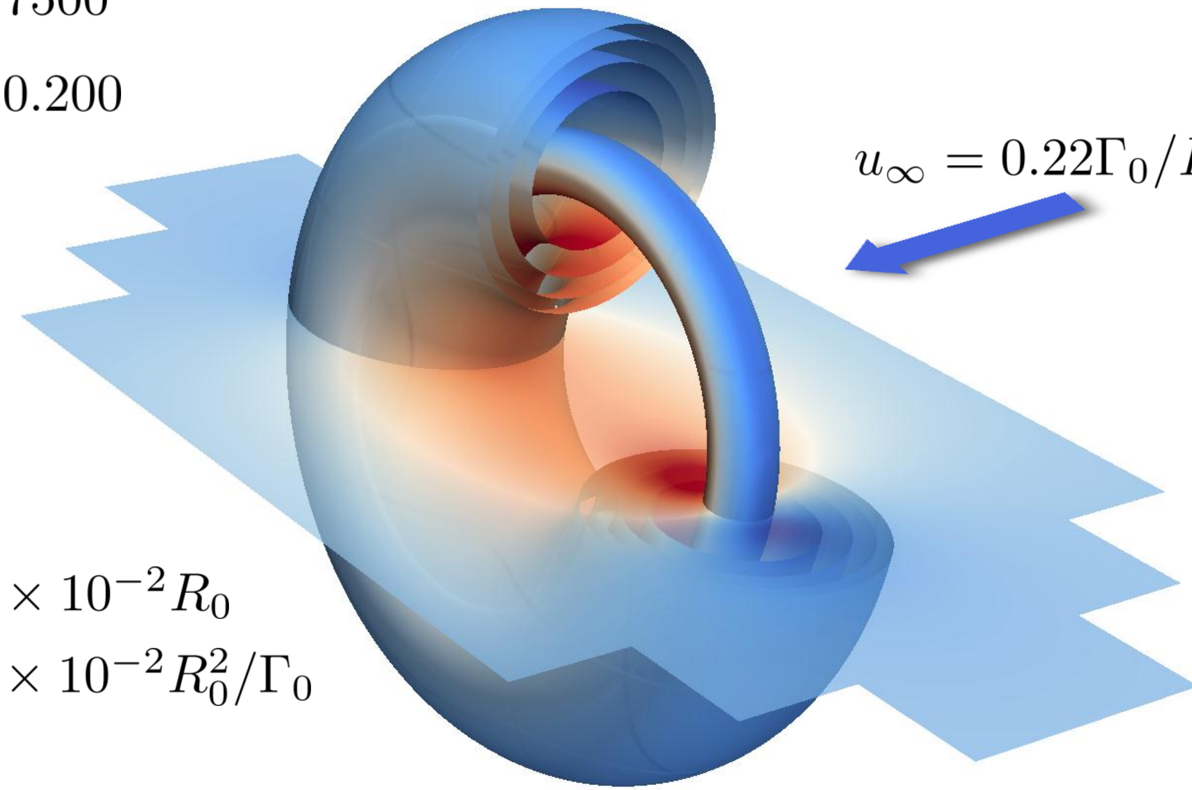
unpack answer

Example: Thin Vortex Ring

$\mathrm{Re} = \Gamma_0/\nu = 7500$

$\delta_0/R_0 = 0.200$

$u_\infty = 0.22\Gamma_0/R_0$

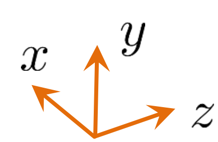


$\Delta x = 1.74 \times 10^{-2} R_0$

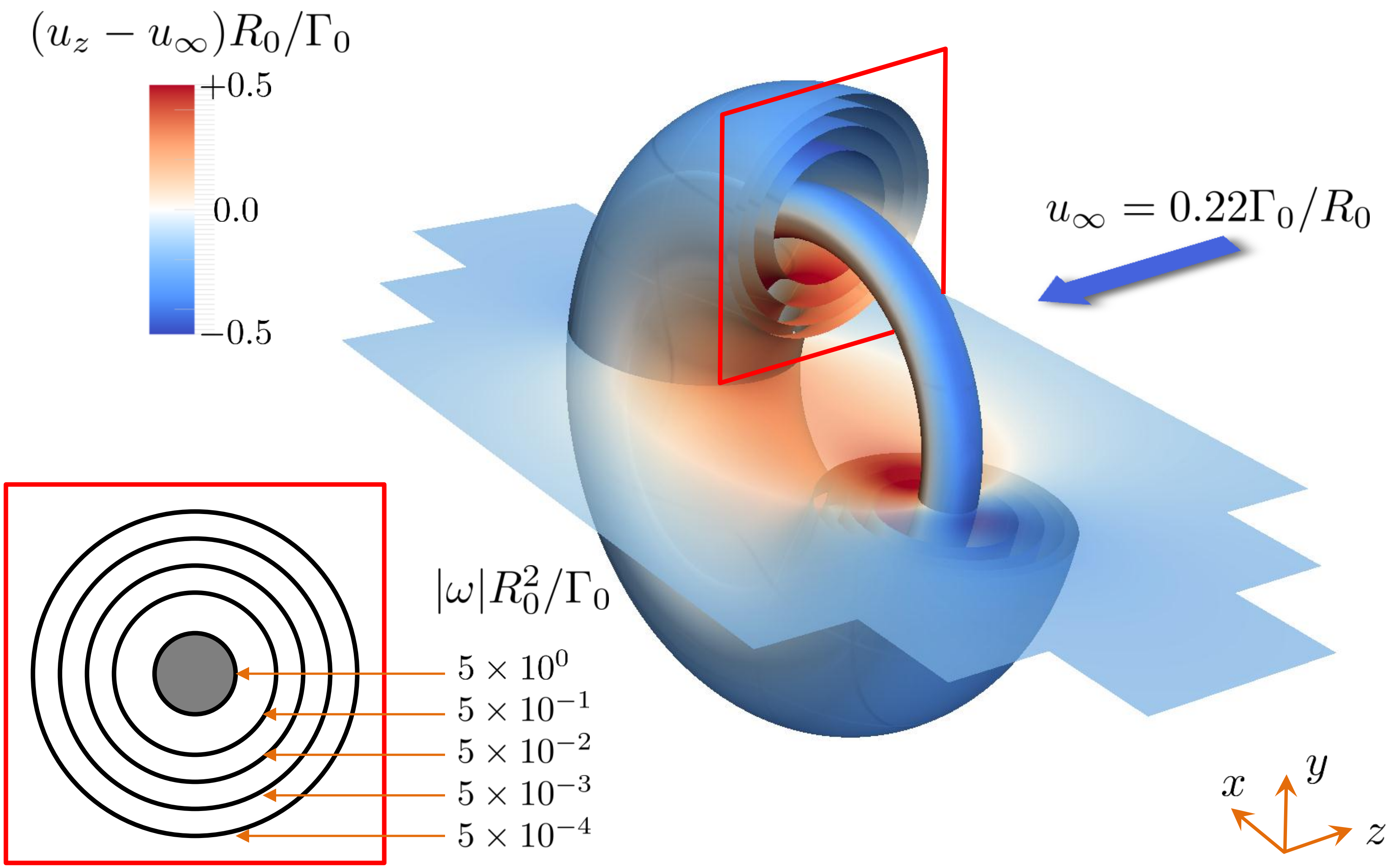
$\Delta t = 5.00 \times 10^{-2} R_0^2/\Gamma_0$

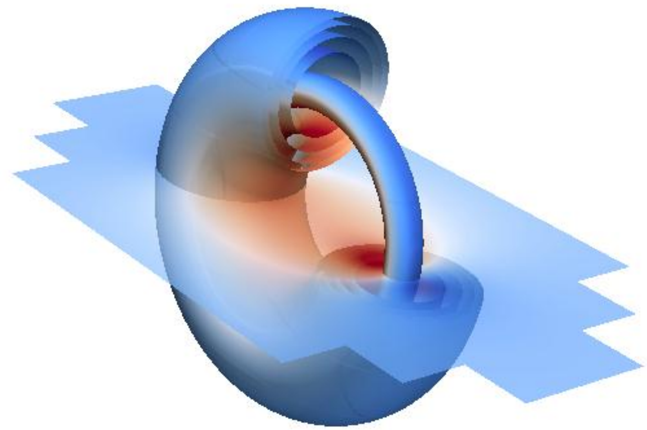
$\mathrm{CFL}_{\max} = 2.71$

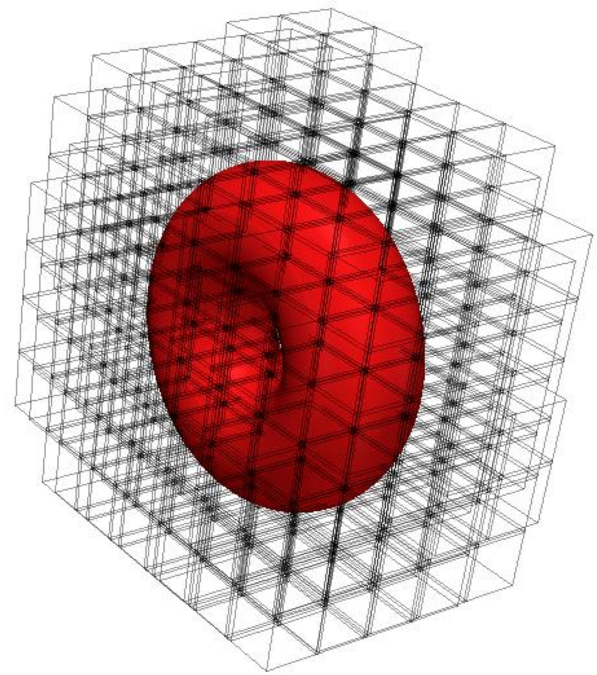
$\alpha = 5.53 \times 10^{-3}$



Example: Thin Vortex Ring

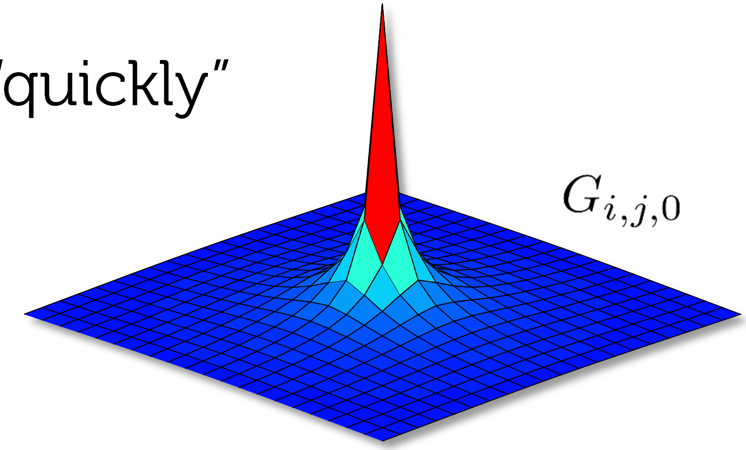


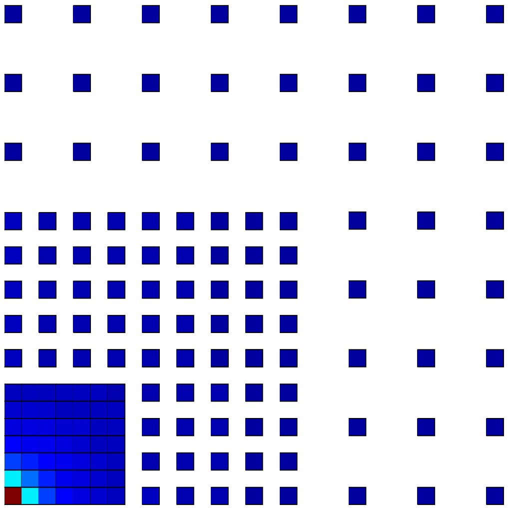




Poisson Solver: Tree-Algorithm

Away from the origin, the LGF “quickly” decays and becomes smooth

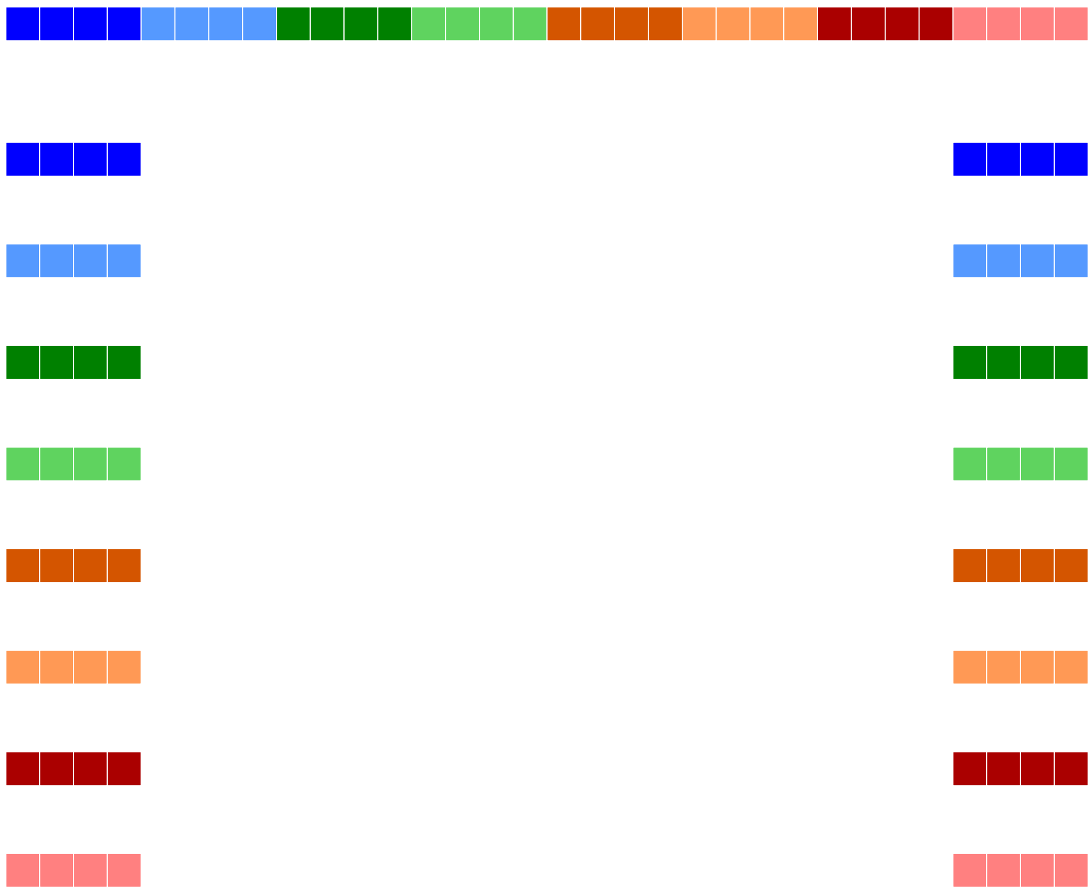




Information generated by LGF can be compressed away from the source

Poisson Solver: Tree-Algorithm

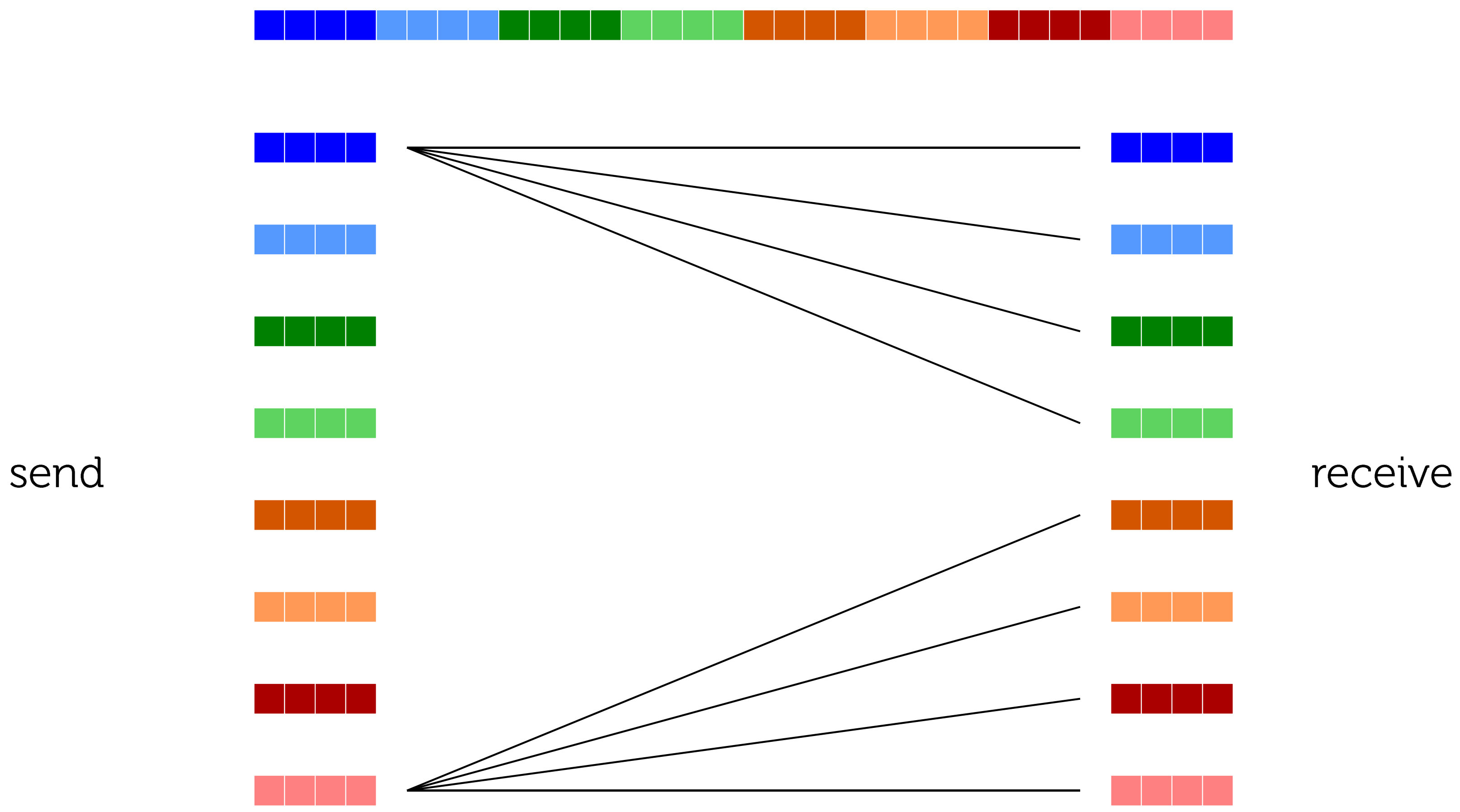
Poisson Solver: Tree-Algorithm



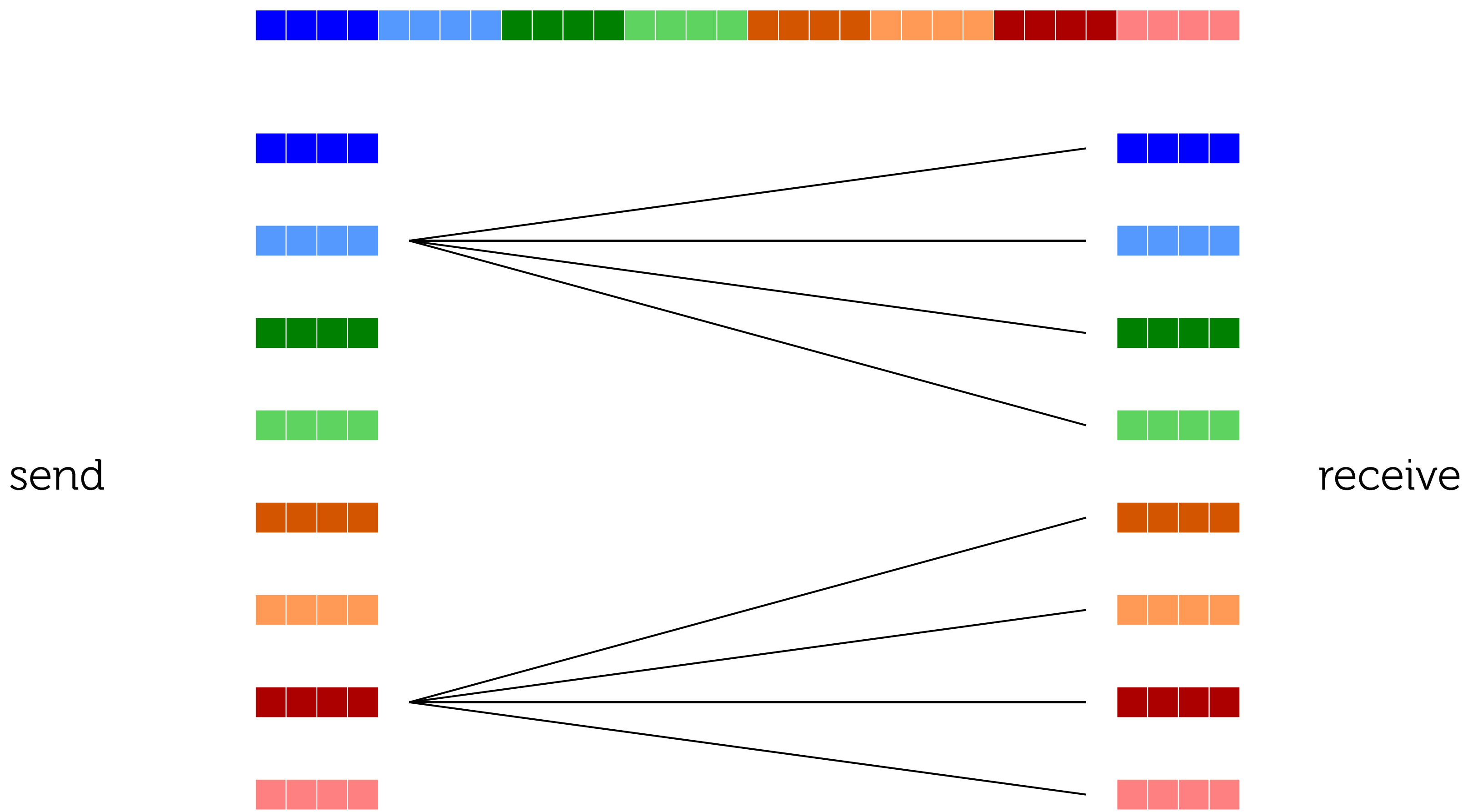
send

receive

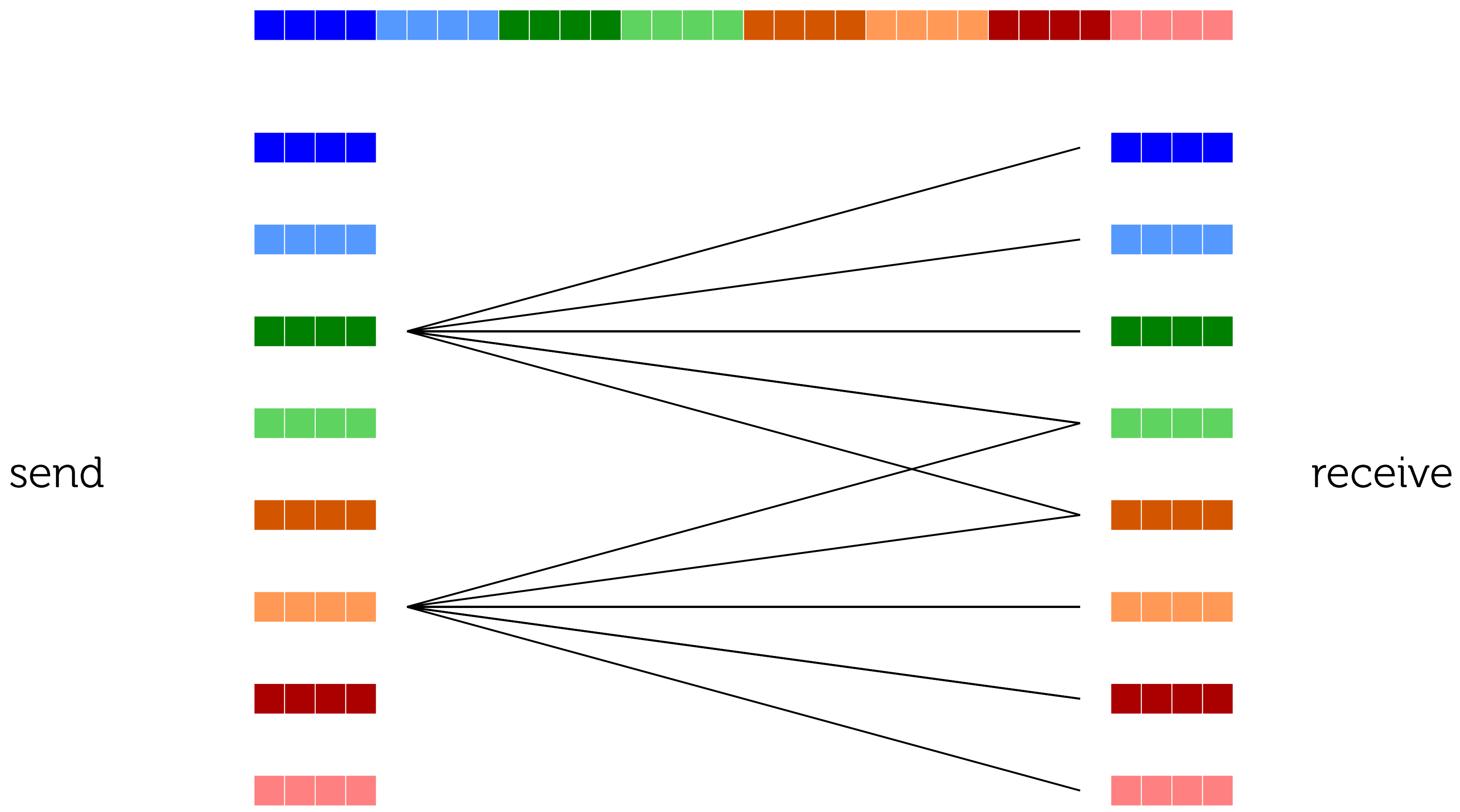
Poisson Solver: Tree-Algorithm



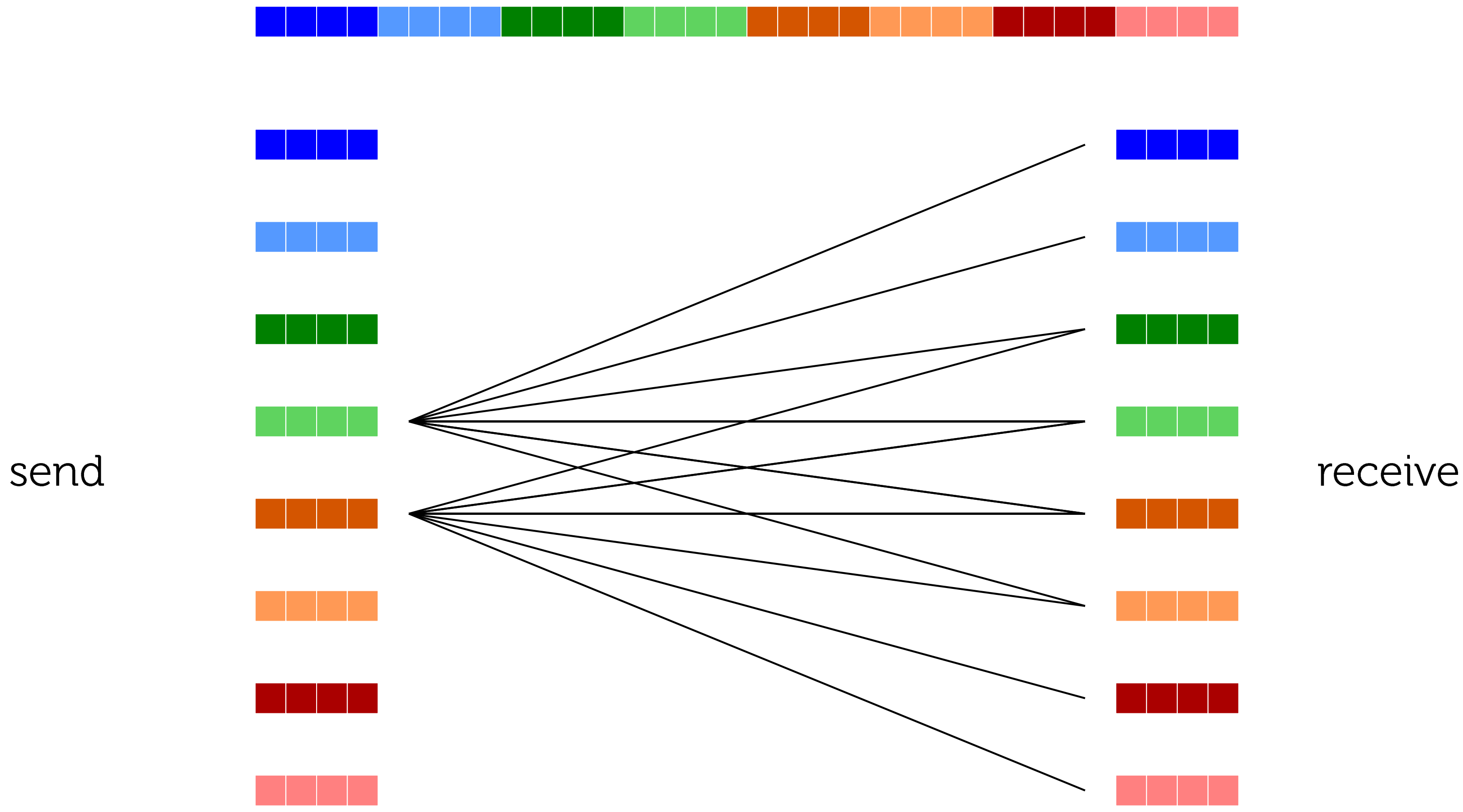
Poisson Solver: Tree-Algorithm



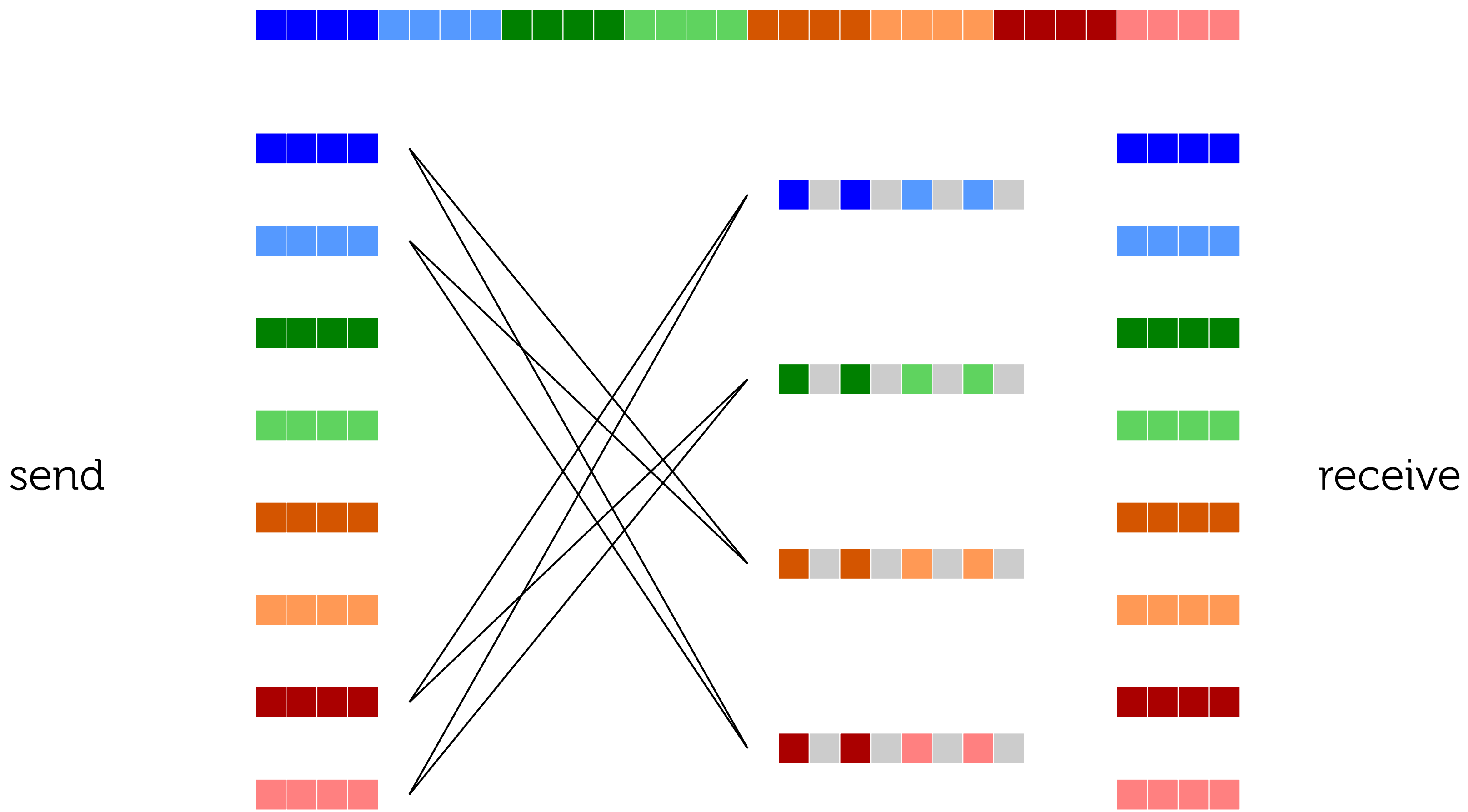
Poisson Solver: Tree-Algorithm



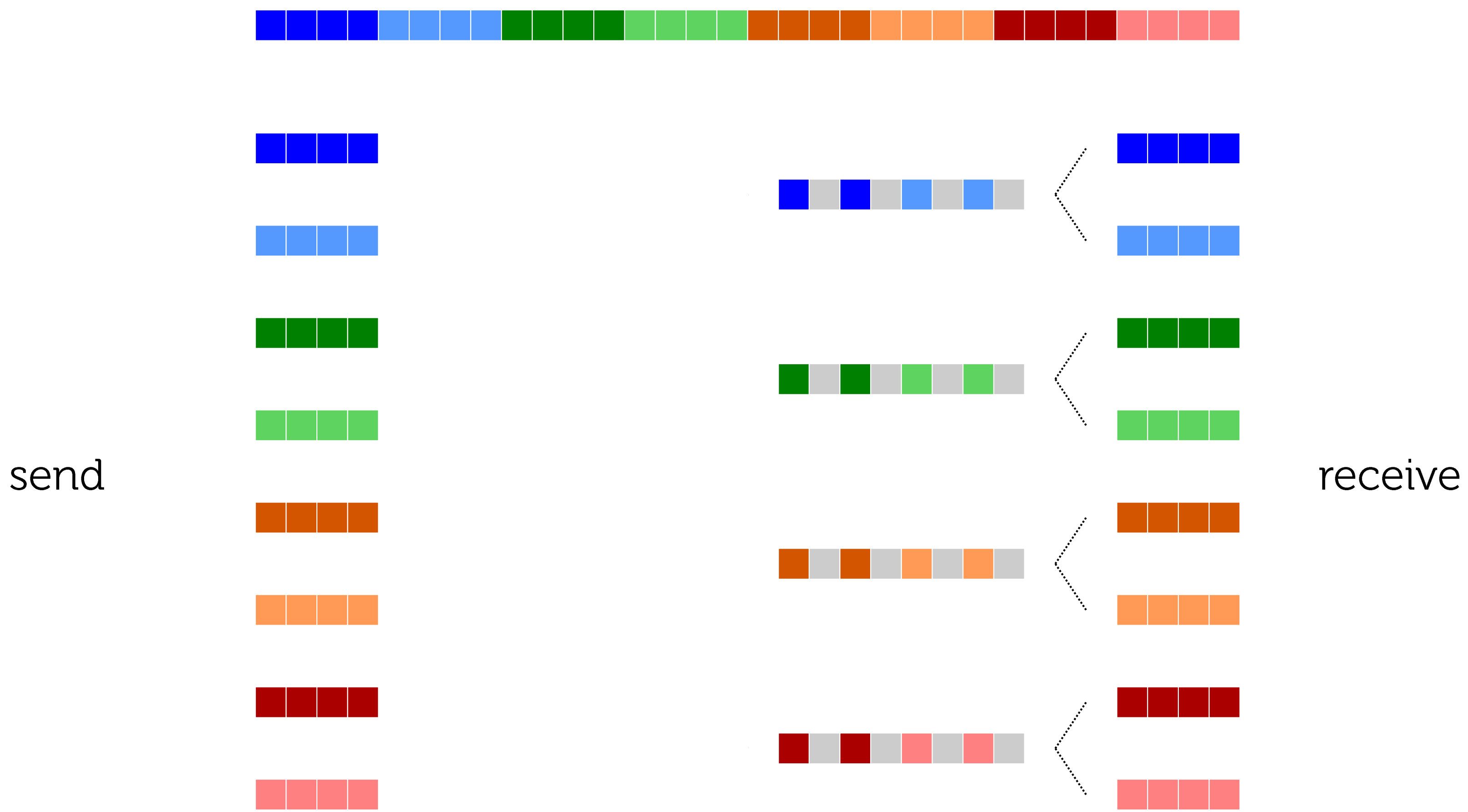
Poisson Solver: Tree-Algorithm



Poisson Solver: Tree-Algorithm



Poisson Solver: Tree-Algorithm



Conclusions

- Incompressible Navier-Stokes solved using novel Lattice Green's Function method
- Solving Poisson problems is the most expensive operation
  - Current solver based on a direct block-to-block interactions
  - Goal for this term is to exploit smoothness of LGF far away from source using tree-like algorithm to compress information

Questions?