Fast Generation of Variable Density Node Distributions for Meshfree Methods

Jure Slak, Gregor Kosec

41st BEM/MRM 2018, 11. 9. 2018

"Jožef Stefan" Institute

Scope

JSI

1. Node generation algorithm requirements
2. Improvements of a published algorithm
3. New algorithm proposition and comparison
4. Numerical examples

Node generation

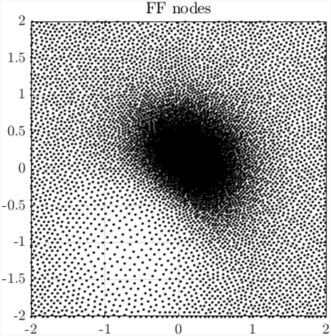
Node generation is a simpler problem than mesh generation.

Why are node generation algorithms needed?

- strong form meshless methods are sensitive to node positioning
- variable density node distributions for adaptivity

Requirements:

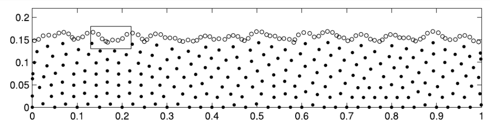
- Input: domain Ω, nodal spacing function $\delta r(p)$ represents approximate distance between p and its neighbors
- Output: N nodes, locally regular
- Works with irregular domains
- Dimension and direction independent
- Minimal spacing guarantees

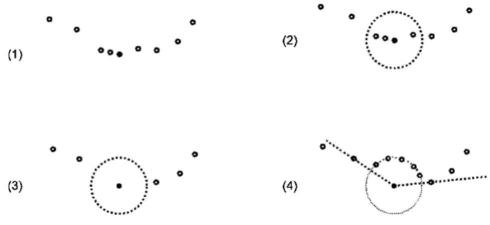


A node positioning algorithm by Fornberg & Flyer

JSI

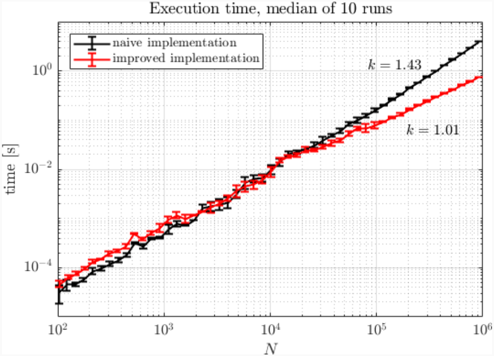
- Given domain Ω, compute a bounding box
- Fill the bounding box with an advancing front algorithm
- Superimpose boundary nodes, discard nodes outside of Ω
- Regularize





Algorithm and figures from: Fornberg, B. and Flyer, N. *Fast generation of 2-D node distributions for mesh-free PDE discretizations.* Computers & Mathematics with

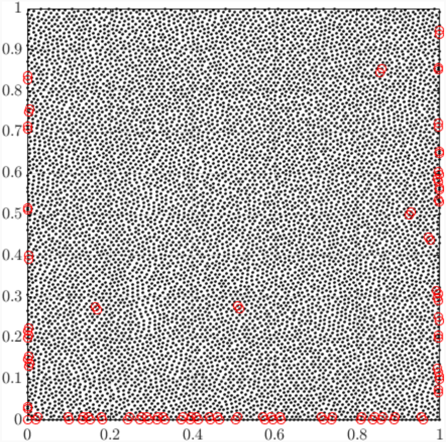
- Naive implementation has time complexity $O(NS)$
- For constant nodal spacing h this is $O(1/h^3)$.
- Improve to $O(N \log S)$ using fast minimum extraction.



Linked list, priority queue and lazy removal allow for $O(\log S)$ amortized minimum search, $O(1)$ access, removal and insertion.

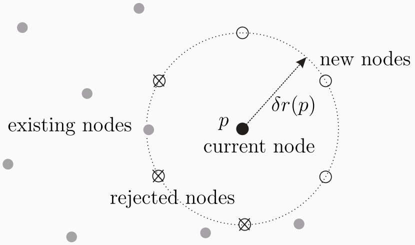
- Split the space as $\mathbb{R}^{d-1} \times \mathbb{R}$ and advance the front along the last coordinate.
- Generating new points: Cartesian products of uniformly angle-spaced point in each dimension
- Efficient implementation generalizes as well:
  use a range search structure (e.g. $(d-1)$-d tree) with a priority queue

- Fills the whole bounding box
- Directionally dependent
- Difficult to implement in 3D
- Does not consider boundary discretization
- No minimal spacing guarantees
- $\|p_i - p_j\| \geq \min \delta r$ violated → see Figure



Poisson disk sampling based algorithm

- Start with a queue of given boundary discretization and additional “seed nodes”.
- In each iteration dequeue a new node p and generate new candidates at a distance $\delta r(p)$.
- Insert acceptable new candidates into the queue.
- Repeat until queue is empty.



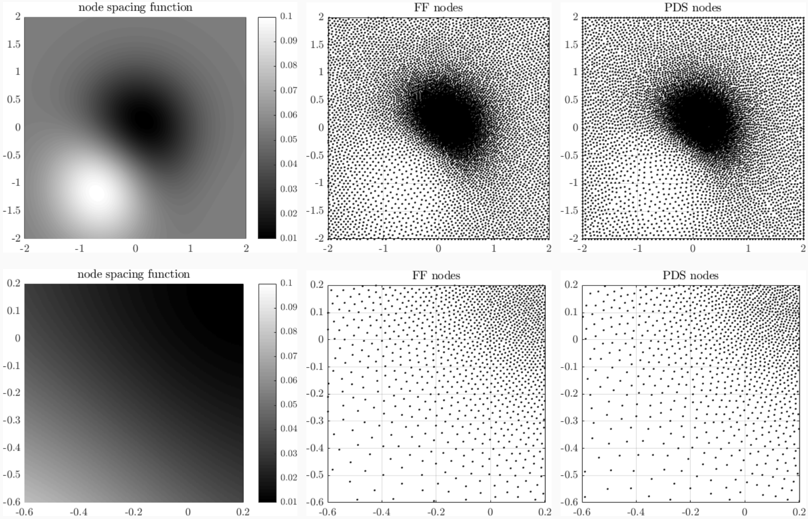
Time complexity: $O(N \log N)$, provable minimal spacing guarantees

Input: Domain Ω and its dimension d. Supports queries of type "Is $p \in \Omega$?".

Input: A nodal spacing function $h \colon \Omega \subset \mathbb{R}^d \to (0, \infty)$.

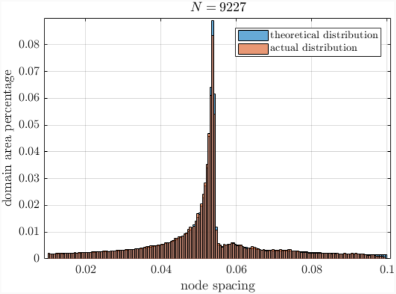
Input: A list of starting points $\mathcal{X}$, this includes the possible boundary discretisation and seed nodes. **Output:** A list of points in Ω distributed according to spacing function h.

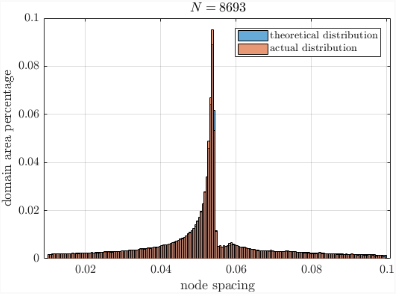
function PDS(Ω, h, $\mathcal{X}$)
    $T \leftarrow$ kdtree_init($\mathcal{X}$) ▷ Initialize spatial search structure on points $\mathcal{X}$.
    $i \leftarrow 0$ ▷ Current node index.
    while $i < |\mathcal{X}|$ **do** ▷ Until the queue is not empty.
        $p_i \leftarrow \mathcal{X}[i]$ ▷ Dequeue current point.
        $r_i \leftarrow h(p_i)$ ▷ Compute its nodal spacing.
        for each $c_{i,j}$ **in** candidates(p_i, r_i) **do** ▷ Generate and loop through candidates.
            if $c_{i,j} \in \Omega$ **then** ▷ Discard candidates outside the


node spacing function
FF nodes
PDS nodes
node spacing function
FF nodes
PDS nodes
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01

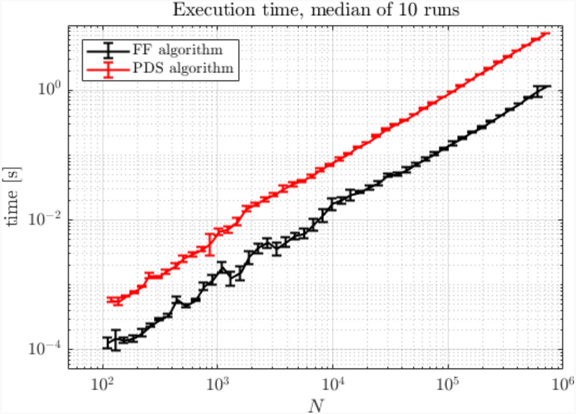
Histograms of internodal distances:

FF PDS





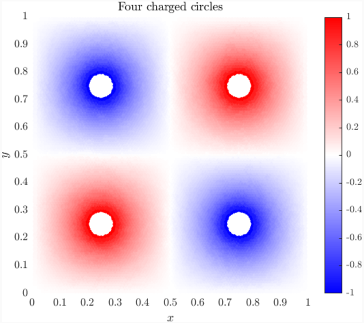
Algorithm was tested on an annulus domain covering approximately 0.5 area of its bounding box. No regularization was applied to FF.

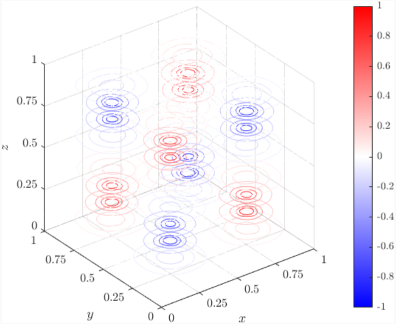


Electrostatics:

$$\nabla^2 \phi = -\rho/\varepsilon$$

Solve for ϕ using RBF-FD in 2D and 3D:





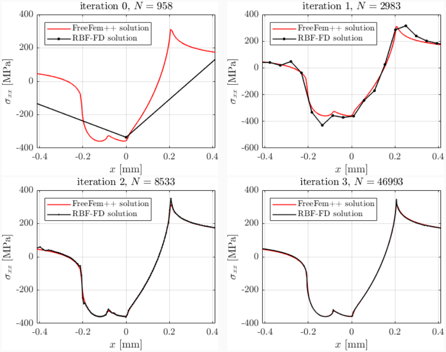
Lamé-Navier equation

$$(\lambda + \mu)\nabla(\nabla \cdot \vec{u}) + \mu \nabla^2 \vec{u} = \vec{f}$$

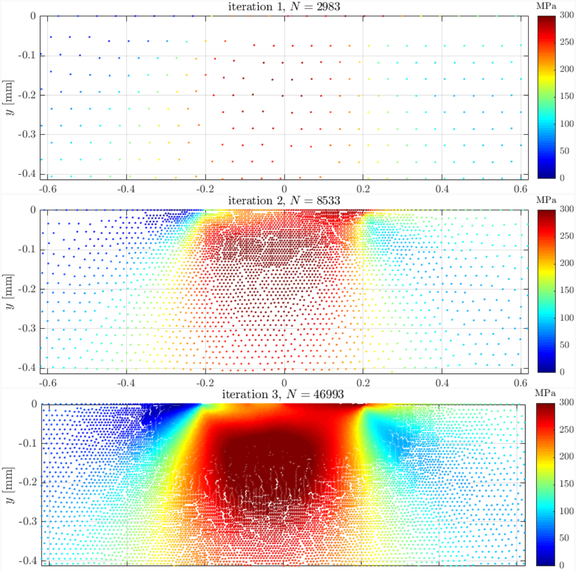
describing displacements $\vec{u} = (u, v)$ and stresses σ.

Case arising from fretting fatigue:

A pad is sliding along and pushing on the specimen. Surface traction is of interest.



Adaptivity in linear elasticity



All computations were done using open source Medusa library.

Medusa

Coordinate Free Mehless Method implementation

`http://e6.ijs.si/medusa/`

Thank you for your attention!

Acknowledgments: FWO Lead Agency project: G018916N Multi-analysis of fretting fatigue using physical and virtual experiments and the ARRS research