Adaptive RBF-FD method for Poisson's equation

Jure Slak, Gregor Kosec

"Jožef Stefan" Institute, Parallel and Distributed Systems Laboratory

2. 7. 2019, BEM/MRM 42

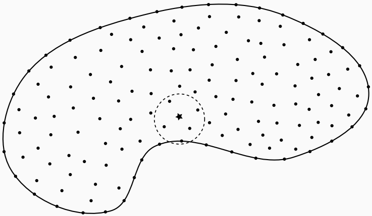
Adaptive procedure:

Discretize domain Ω to obtain discretization $\mathcal{X}^{(0)}$
For $j = 0, \dots, I_{max}$

1. Solve the problem and obtain $u^{(j)}$
2. Estimate errors $e^{(j)}$
3. If $\|e^{(j)}\| < \varepsilon$, return $u^{(j)}$
4. Refine the discretization $\mathcal{X}^{(j)}$ using $e^{(j)}$ to obtain $\mathcal{X}^{(j+1)}$

Domain discretization:

- Points x_i on the boundary and in the interior
- Point neighborhoods $N(x_i)$



Classical Finite Differences:

$$u''(x_i) \approx \frac{1}{h^2}u(x_{i-1}) - \frac{2}{h^2}u(x_i) + \frac{1}{h^2}u(x_{i+1})$$

Generalized Finite Differences:

$$(\mathcal{L}u)(x_i) \approx \sum_{x_j \in N(x_i)} w_j^i u(x_j)$$

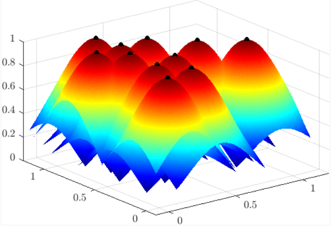
+ exactness for a certain set of functions (e.g. monomials)

Solution procedure – RBF-FD

JSI

Exactness is imposed for Radial Basis Functions

- Given nodes $X = \{x_1, \ldots, x_n\}$ and a radial function $\varphi = \varphi(r)$
- Generate $\{\varphi_i := \varphi(\|\cdot - x_i\|), x_i \in X\}$



Imposing exactness of

$$(\mathcal{L}u)(x_i) \approx \sum_{x_j \in N(x_i)} w_j^i u(x_j)$$

for each φ_j for $x_j \in N(x_i)$, we get

$$\begin{bmatrix} \varphi(\|x_{j_1} - x_{j_1}\|) & \cdots & \varphi(\|x_{j_{n_i}} - x_{j_1}\|) \\ \vdots & \ddots & \vdots \\ \varphi(\|x_{j_1} - x_{j_{n_i}}\|) & \cdots & \varphi(\|x_{j_{n_i}} - x_{j_{n_i}}\|) \end{bmatrix} \begin{bmatrix} w_{j_1}^i \\ \vdots \\ w_{j_{n_i}}^i \end{bmatrix} = \begin{bmatrix} (\mathcal{L}\varphi_{j_1})(x_i) \\ \vdots \\ (\mathcal{L}\varphi_{j_{n_i}})(x_i) \end{bmatrix}$$

Enforce consistency up to certain order, e.g. for constants

$$\begin{bmatrix} A & \mathbf{1} \\ \mathbf{1}^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_\varphi \\ 0 \end{bmatrix}$$

In general:

$$\begin{bmatrix} A & P \\ P^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_\varphi \\ \boldsymbol{\ell}_p \end{bmatrix},$$

where

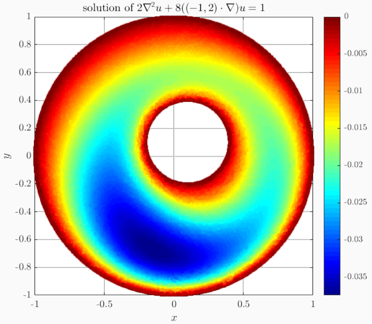
$$P = \begin{bmatrix} p_1(\boldsymbol{x}_1) & \cdots & p_s(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_n) & \cdots & p_s(\boldsymbol{x}_n) \end{bmatrix}, \quad \boldsymbol{\ell}_p = \begin{bmatrix} (\mathcal{L}p_1)|_{\boldsymbol{x}=\boldsymbol{x}^*} \\ \vdots \\ (\mathcal{L}p_s)|_{\boldsymbol{x}=\boldsymbol{x}^*} \end{bmatrix}.$$

Solution procedure – overview

JSI

Problem:

$$\mathcal{L}u = f \quad \text{on } \Omega,$$
$$u = u_0 \quad \text{on } \partial\Omega,$$



1. Discretize domain Ω
2. Find neighborhoods $N(x_i)$
3. Compute weights $\boldsymbol{w}^i$ for approximation of $\mathcal{L}$ over $N(x_i)$
4. Assemble weights in a sparse system $Wu = f$
5. Solve the sparse system $Wu = f$
6. Approximate/interpolate the solution

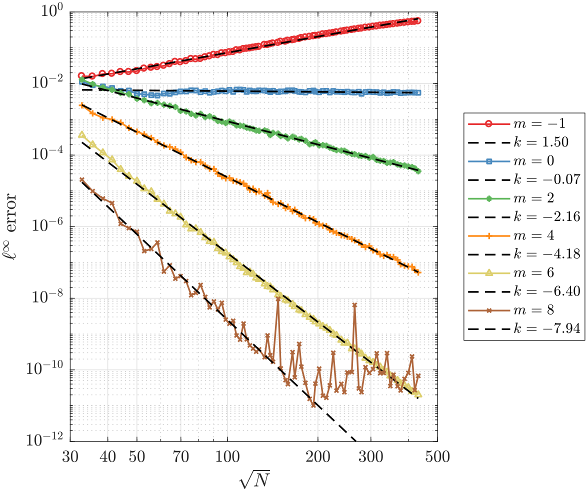
Test case: Poisson problem

JSI

Annulus domain, scattered nodes

convergence orders match augmentation

ℓ_1 and ℓ_2 errors similar

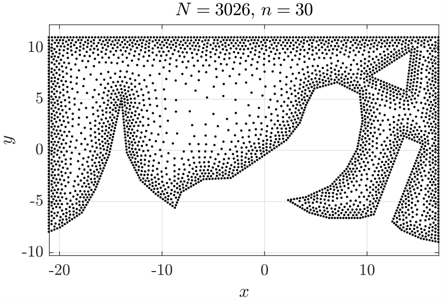


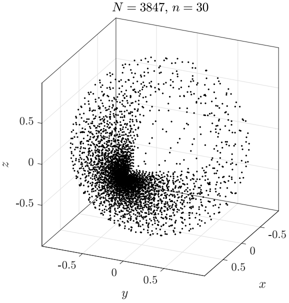
Fill domain Ω with locally regular nodes according to arbitrary spacing function $h(p)$.

Recent algorithms for variable density node positioning in 3D:

Slak & Kosec, 2018: `https://arxiv.org/abs/1812.03160`

van der Sande & Fornberg, 2019: `https://arxiv.org/abs/1906.00636`





Error indicator and refinement

JSI

Simple error indicator:

$$\hat{u}(x_i) = \frac{1}{n} \sum_{x_j \in N(x_i)} u(x_j)$$

$$e_i^2 = \frac{1}{n} \sum_{x_j \in N(x_i)} |\hat{u}(x_i) - u(x_j)|^2$$

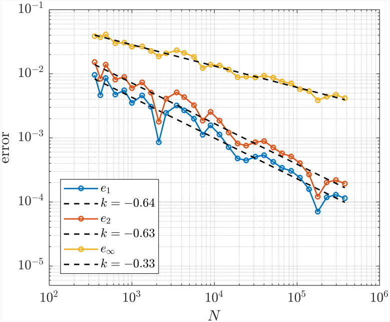
Refinement by appropriate modification of h. Three cases:

$$\begin{cases} \text{increase} & \text{if } e_i > \varepsilon \\ \text{no change} & \text{if } \eta \le e_i \le \varepsilon \\ \text{decrease} & \text{if } e_i < \eta \end{cases}$$

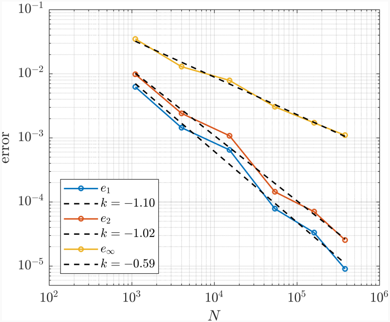
Increase/decrease proportional to e_i, maximal change for factor α.

L-shaped domain – convergence

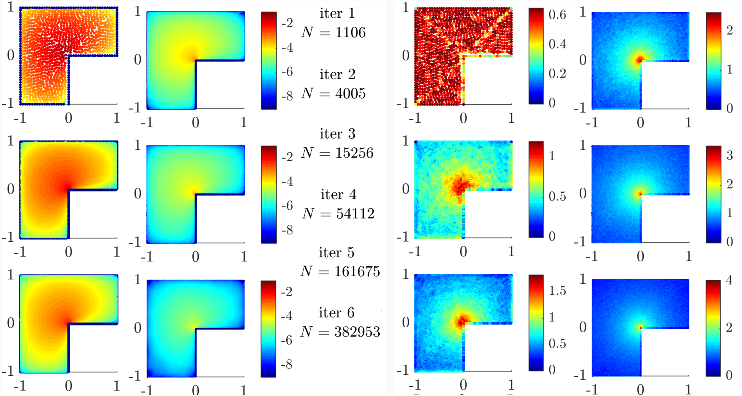
Uniform



Adaptive

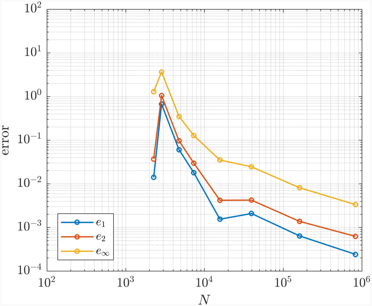


Adaptive iteration: error and node density $\rho = -\log_2 \frac{d_i}{\max_j d_j}$.

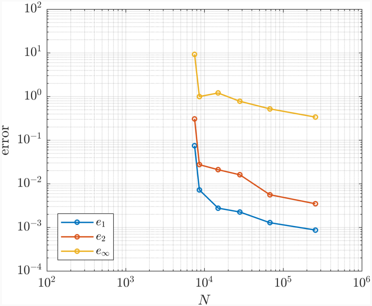


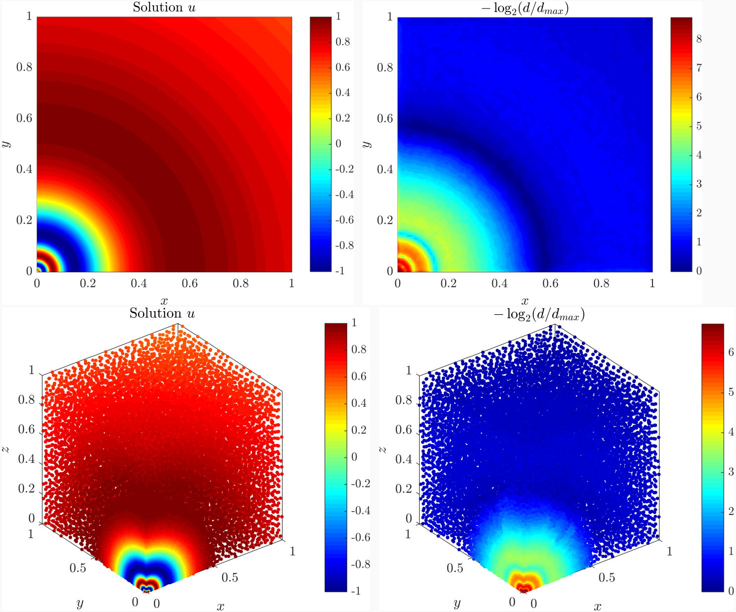
$$-\nabla^2 u + \frac{1}{(r+\alpha)^4} = f, \qquad u(r) = \sin\left(\frac{1}{\alpha + r}\right)$$

2D



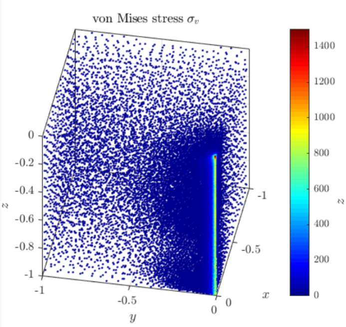
3D

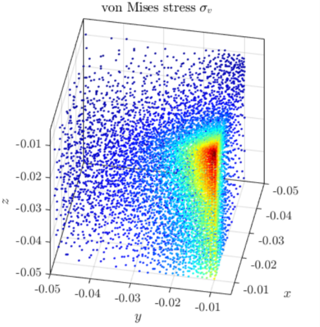




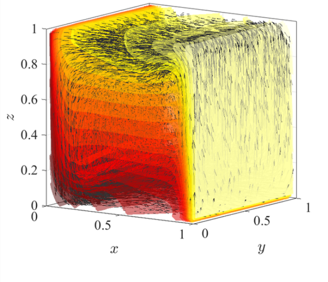
Solution u
$-\log_2(d/d_{max})$
Solution u
$-\log_2(d/d_{max})$

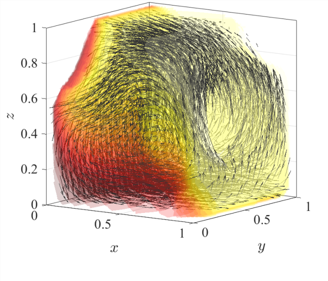
Additional examples



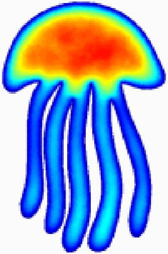


Support for: complex numbers, ghost nodes, coupled domains





All computations were done using open source Medusa library.



Medusa

Coordinate Free Mehless Method implementation

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

Slides available at http://e6.ijs.si/~jslak/.

Thank you for your attention!

Acknowledgments: FWO Lead Agency project: G018916N Multi-analysis of fretting fatigue using physical and virtual experiments, the ARRS research core funding No. P2-0095 and Young Researcher program PR-08346.