Adaptive RBF-FD method (Adaptivna RBF-FD metoda)

Jure Slak

Presentation of thesis results before the final defence

8. 7. 2020

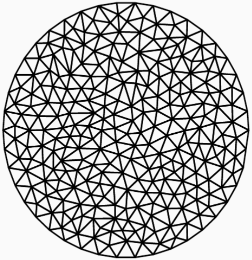
Overview

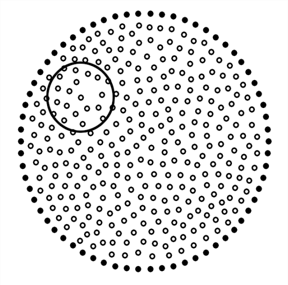
JSI

- Thesis repo (includes this presentation):
  `https://gitlab.com/jureslak/phd`
- Thesis overview
  1. RBFs and their properties
  2. Local strong form operator approximations
  3. Domain discretization generation
  4. Fully automatic adaptivity
  5. Implementation
- Main results (chapters 3, 4, and 5)
  - Node generation algorithms
  - h-adaptivity for RBF-FD
  - Implementation

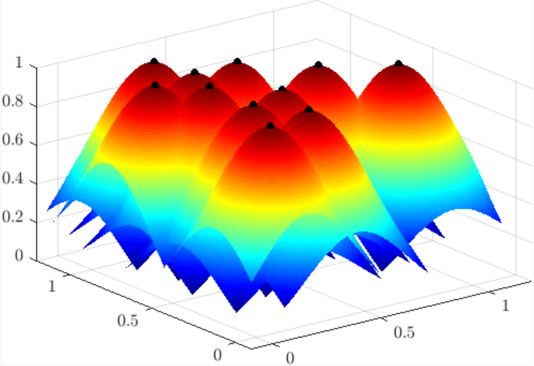
Classification of RBF-FD

	mesh/grid-based	meshless
strong-form	FDM	FPM, RBF-FD, GFDM
weak form	FEM, IGA BEM	EFG, MLPG





RBF: function of form $\varphi(\|x\|)$



[Wen04] H. Wendland, *Scattered data approximation*, Cambridge Monographs on Applied and Computational Mathematics **17**, Cambridge University Press, 2004

Strong form approximations:

$$(\mathcal{L}u)(p) \approx \sum_{i=1}^{n} w_i u(p_i)$$

Enforce exactness for a class of functions:

$$\begin{bmatrix} \varphi(\|p_1 - p_1\|) & \cdots & \varphi(\|p_n - p_1\|) \\ \vdots & \ddots & \vdots \\ \varphi(\|p_1 - p_n\|) & \cdots & \varphi(\|p_n - p_n\|) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \mathcal{L}\varphi(\|p - p_1\|) \\ \vdots \\ \mathcal{L}\varphi(\|p - p_n\|) \end{bmatrix},$$

Obtain $\boldsymbol{w}_{\mathcal{L},p} \approx \mathcal{L}|_p$.

Solvable? **Yes!**

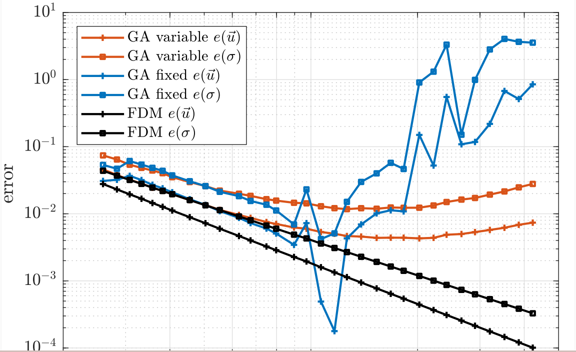
RBF-FD method: accuracy and conditioning

Sometimes computable in closed form:

$$w = \left\{ \frac{4h^2 e^{\frac{3h^2}{\sigma^2}}}{\sigma^4\left(e^{\frac{2h^2}{\sigma^2}}-1\right)^2}, -\frac{2\left(\sigma^2 e^{\frac{4h^2}{\sigma^2}} + e^{\frac{2h^2}{\sigma^2}}\left(4h^2-2\sigma^2\right)+\sigma^2\right)}{\sigma^4\left(e^{\frac{2h^2}{\sigma^2}}-1\right)^2}, \frac{4h^2 e^{\frac{3h^2}{\sigma^2}}}{\sigma^4\left(e^{\frac{2h^2}{\sigma^2}}-1\right)^2} \right\}$$

Approximation error:

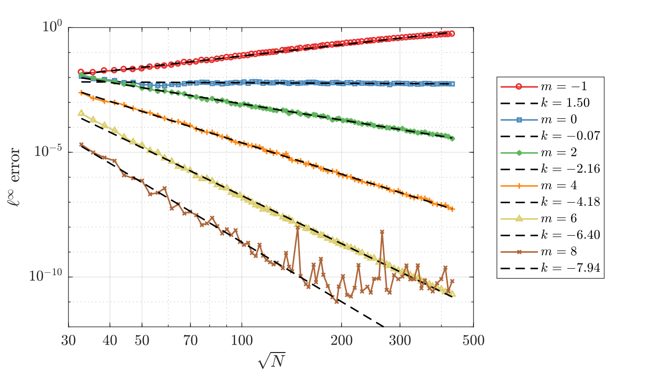
$$w^\mathsf{T}\boldsymbol{u} = u''(x_0) + h^2\left(\frac{u(x_0)}{\sigma^4} + \frac{u''(x_0)}{\sigma^2} + \frac{1}{12}u^{(4)}(x_0)\right) + O\left(h^3\right)$$



RBF-FD method: accuracy and conditioning

Some proposed solutions:

- Stable computation of $\boldsymbol{w}_{\mathcal{L},p}$
- Polyharmonic RBF $\varphi(r) = r^k$ and monomial augmentation



Challenges: computational time, stability wrt. nodal positioning

The goal

Develop a fully automatic adaptive solution procedure for RBF-FD.

1. Discretize the domain
2. Solve the problem
3. Estimate the error
4. If the error is below threshold, return the solution. Else:
5. Refine the discretization
6. Go to 2

Many different ways of estimation and refinement.

Our choice: h-refinement, "re-meshing" approach, gradient-based error indicators

Domain discretization

Domain discretization: boundary nodes + internal nodes + stencils

Stencils: k closest nodes

Domain generation algorithm requirements:

- Sufficient quality
- Variable density
- 2D and 3D (and more?)
- Boundaries and interiors
- Computational complexity

Result: new algorithms for node generation

J. Slak and G. Kosec, *On generation of node distributions for meshless PDE discretizations*, SIAM J. Sci. Comput. 41(5):A3202–A3229, 2019.

U. Duh, G. Kosec, and J. Slak, *Fast variable density node generation on parametric surfaces with application to mesh-free methods*, arXiv:2005.08767,

For uniform nodes:

Definition (Fill distance)

$$h_{X,\Omega} = 2 \sup_{\boldsymbol{x} \in \Omega} \min_{j=1,\dots,n} \|\boldsymbol{x} - \boldsymbol{x}_j\|$$

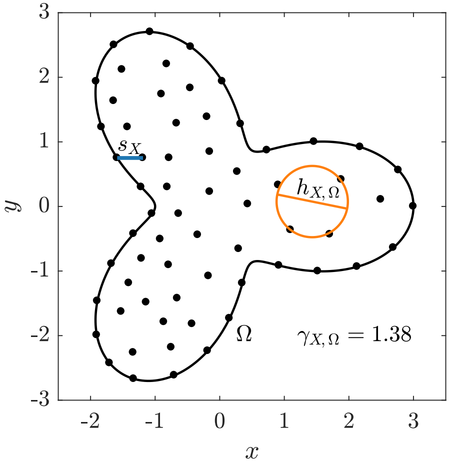
Definition (Separation distance)

$$s_X = \min_{1 \le i < j \le n} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|.$$

Definition (Node set ratio)

$$\gamma_{X,\Omega} = \frac{h_{X,\Omega}}{s_X}.$$

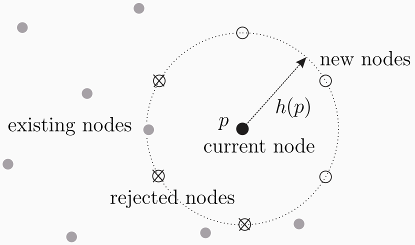
Approximation theorems: convergence: $h_{X,\Omega}$, stability: s_X.



Node generation in domain interior

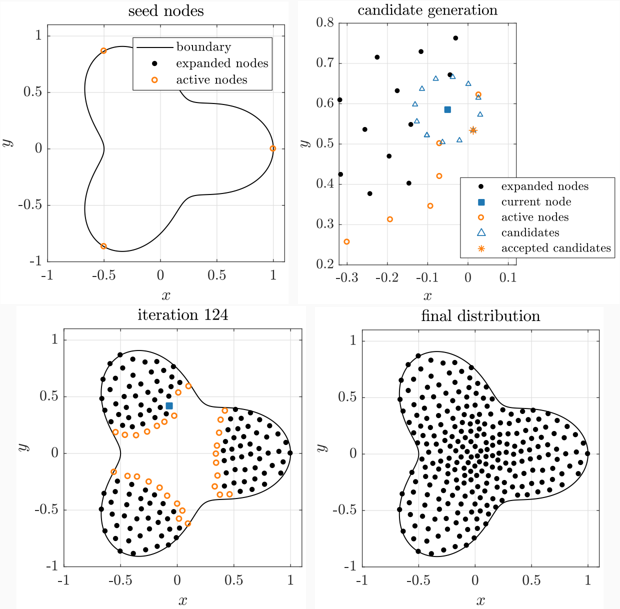
JSI

- Enqueue the “seed nodes”
- In each iteration, remove a node p and generate candidates at a radius $h(p)$.
- Enqueue the accepted candidates
- Repeat until the queue is empty



Node generation in domain interior

JSI



Time complexity: using a structure S:

$$T_S(N) = I_S(N) + O(N(T_h + n_c(T_\Omega + Q_S(N)))) + NI_S(N)$$

k-d tree: $O(N \log N)$, k-d grid (uniform data): $O(N)$

Finiteness: Bounded domain and positive h are not enough. h must be bounded away from 0.

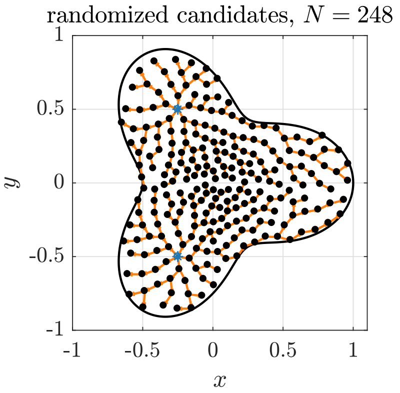
Assume seed notes are valid.

Constant h:

$$\|p_i - p_j\| \geq h$$

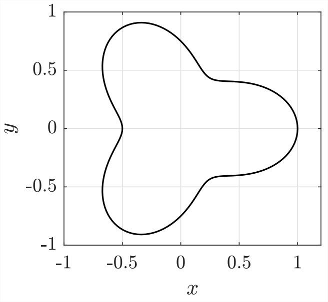
Theorem (Minimal spacing guarantee)

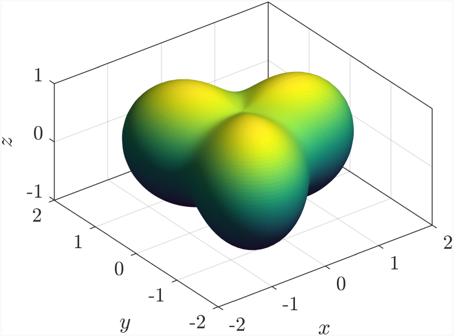
$$\|p_i - p_j\| \geq h(p_{\beta(j)})$$



Other options h:

$$\|p_i - p_j\| \geq \max(h_i, h_j), \min(h_i, h_j), h_i, h_j$$

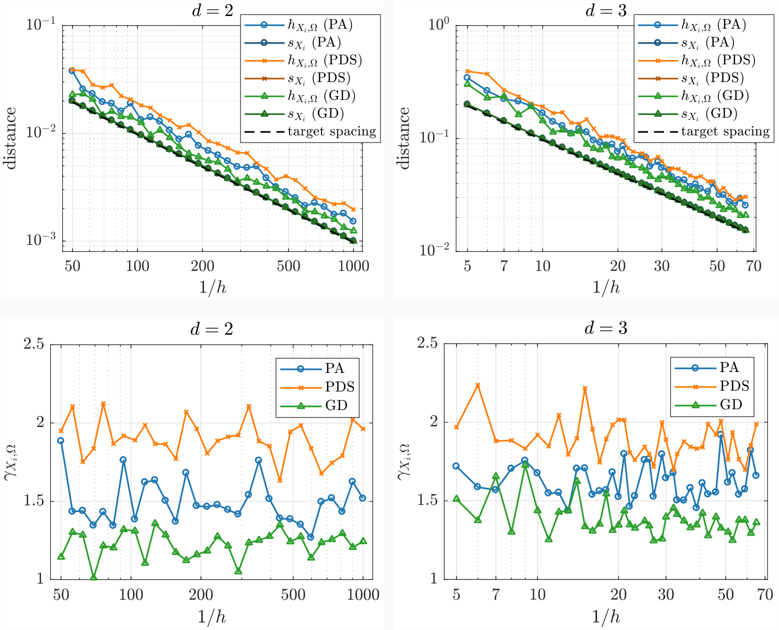




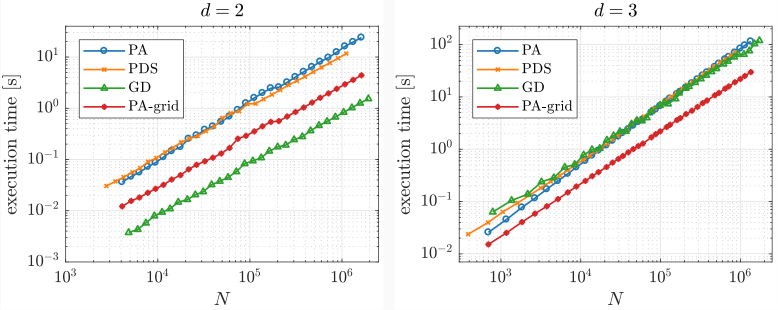
$$r_2(\varphi) = \frac{1}{4}(3 + \cos(3\varphi),\ \ r_3(\varphi, \vartheta) = \frac{1}{4}\left(3 + \frac{\vartheta^2(\pi - \vartheta)^2}{4}(2 + \cos(3\varphi))\right)$$

$$\boldsymbol{r}_2(\varphi) = \begin{bmatrix} r_2(\varphi)\cos(\varphi) \\ r_2(\varphi)\sin(\varphi) \end{bmatrix},\ \ \boldsymbol{r}_3(\varphi, \vartheta) = \begin{bmatrix} r_3(\varphi, \vartheta)\sin(\vartheta)\cos(\varphi) \\ r_3(\varphi, \vartheta)\sin(\vartheta)\sin(\varphi) \\ r_3(\varphi, \vartheta)\cos(\vartheta) \end{bmatrix}$$

Quasi-uniformity



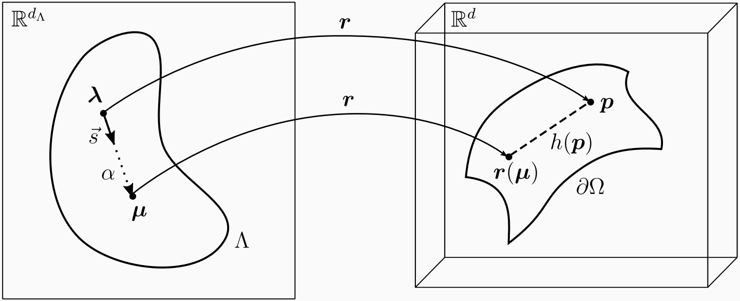
Execution time



Regular parametrization:

$$\boldsymbol{r}\colon \Lambda \subseteq \mathbb{R}^{d_\Lambda} \to \partial\Omega \subseteq \mathbb{R}^d$$

Place parameters, map to points.



$$\boldsymbol{r}(\boldsymbol{\mu}) = \boldsymbol{r}(\boldsymbol{\lambda} + \alpha\vec{s}) = \boldsymbol{r}(\boldsymbol{\lambda}) + \alpha\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s} + \boldsymbol{R}(\boldsymbol{\lambda}, \alpha, \vec{s})$$

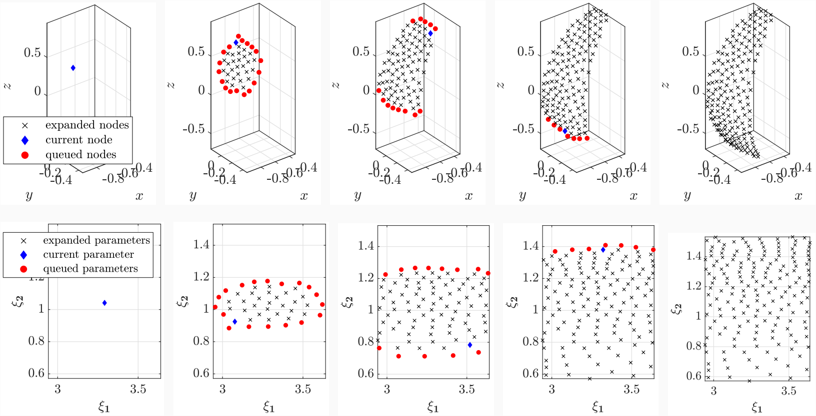
Finding the distance α:

$$h(\boldsymbol{r}(\boldsymbol{\lambda})) = \|\boldsymbol{r}(\boldsymbol{\lambda}) - \boldsymbol{r}(\boldsymbol{\lambda}) - \alpha\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s}\| = \alpha\|\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s}\|,$$

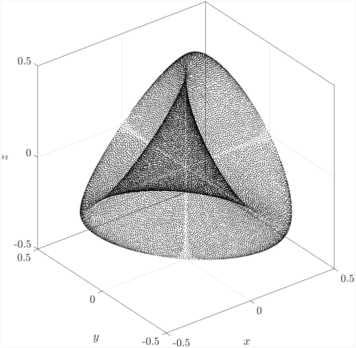
$$\boldsymbol{\mu} = \boldsymbol{\lambda} + \frac{h(\boldsymbol{r}(\boldsymbol{\lambda}))}{\|\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s}\|}\vec{s}, \quad \alpha = \frac{h(\boldsymbol{r}(\boldsymbol{\lambda}))}{\|\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s}\|}.$$

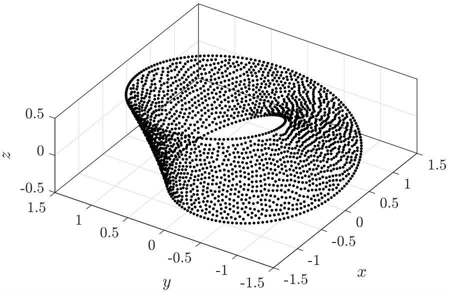
Actual distance:

$$\hat{h}(\boldsymbol{\lambda}, \vec{s}) = \left\|\boldsymbol{r}(\boldsymbol{\lambda}) - \boldsymbol{r}\left(\boldsymbol{\lambda} + \frac{h(\boldsymbol{r}(\boldsymbol{\lambda}))}{\|\nabla\boldsymbol{r}(\boldsymbol{\lambda})\vec{s}\|}\vec{s}\right)\right\|.$$


expanded nodes
current node
queued nodes
z
y
x
0.5
0
-0.5
0
-0.2
-0.4
-0.8
-0.6
-0.4
expanded parameters
current parameter
queued parameters
ξ_1
ξ_2
1.4
1.2
1
0.8
0.6
3
3.5

More general embedded manifolds: non-orientable, co-dimension





$$|\Delta h(\boldsymbol{\lambda}, \vec{s})| \le \max_{\boldsymbol{\lambda}\in\Lambda} \left(\frac{\sqrt{d_\Lambda}}{2} h(\boldsymbol{p})^2 \frac{\max\limits_{i=1,\dots,d_\Lambda} \max\limits_{\boldsymbol{\zeta}\in\bar{B}(\boldsymbol{\lambda},\rho_{\boldsymbol{\lambda}})\cap\Lambda} \sigma_1((\nabla\nabla r_i)(\boldsymbol{\zeta}))}{\sigma_{d_\Lambda}(\nabla \boldsymbol{r}(\boldsymbol{\lambda}))^2} \right)$$

$$\le \frac{\sqrt{d_\Lambda}}{2} h_M^2 \frac{\sigma_{1,M}(\nabla\nabla \boldsymbol{r})}{\sigma^2_{d_\Lambda,m}(\nabla \boldsymbol{r})},$$

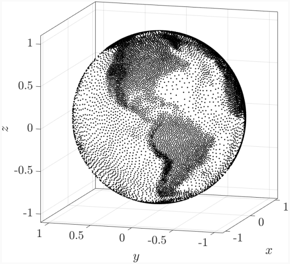
where

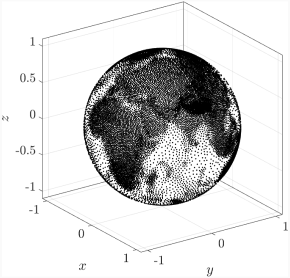
$$h_M^2 = \max_{\boldsymbol{\lambda}\in\Lambda} h(\boldsymbol{r}(\boldsymbol{\lambda}))^2,$$

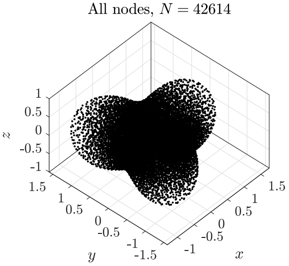
$$\sigma_{1,M}(\nabla\nabla \boldsymbol{r}) = \max_{i=1,\dots,d_\Lambda} \max_{\boldsymbol{\lambda}\in\Lambda} \sigma_1((\nabla\nabla r_i)(\boldsymbol{\lambda})),$$

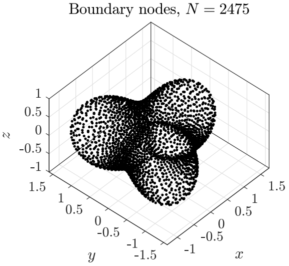
$$\sigma_{d_\Lambda,m}(\nabla \boldsymbol{r}) = \min_{\boldsymbol{\lambda}\in\Lambda} \sigma_{d_\Lambda}(\nabla \boldsymbol{r}(\boldsymbol{\lambda})),$$

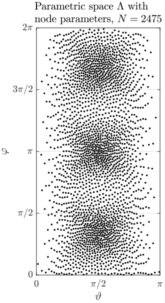
Variable density









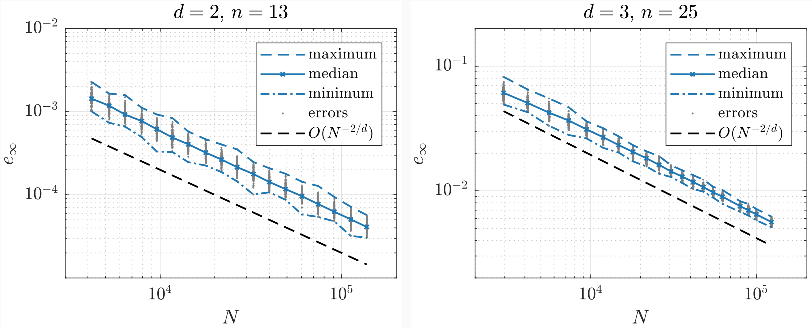


Also explored in the thesis:

- Node quality for surface node placing
- Node quality for variable density distributions
- Better node quality measures
- Execution time for variable density distributions
- Time complexity for surface nodes

One final item: Behavior of the method on generated nodes.

PDE accuracy



Solving a Poisson problem in Ω_2 and Ω_3 using RBF-FD with $\varphi(r) = r^3$, n closest neighbor stencils and uniform discretization. Each gray point is one run.

The node placing algorithm and the method work well together.

With robust node generation, we can move on to adaptivity.

Result: A fully automatic h-adaptive "re-meshing" procedure for RBF-FD.

[SK19a] J. Slak and G. Kosec. *Refined Meshless Local Strong Form solution of Cauchy–Navier equation on an irregular domain* , Engineering analysis with boundary elements **100**:3–13, 2019.

[SK19b] J. Slak and G. Kosec, *Adaptive radial basis function-generated finite differences method for contact problems*, International Journal for Numerical Methods in Engineering **119**(7):661–686, 2019.

Adapt the spacing function, not the discretization.

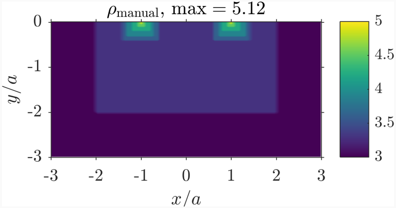
$$h_i^{(j+1)} := \max\{\min\{h_i^{(j)}/f_i^{(j)}, h_{\mathsf{d}}(\boldsymbol{x}_i)\}, h_{\mathsf{r}}(\boldsymbol{x}_i)\},$$

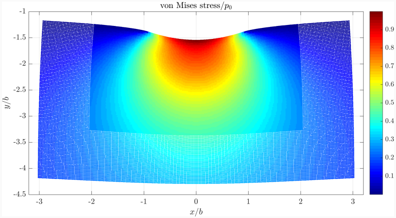
where the density increase factor $f_i^{(j)}$ is defined as

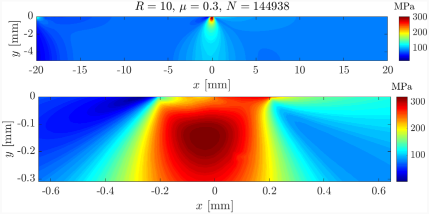
$$f_i^{(j)} = \begin{cases} 1 + \frac{\varepsilon_{\mathsf{d}} - \hat{e}_i^{(j)}}{\varepsilon_{\mathsf{d}} - m^{(j)}}\left(\frac{1}{\alpha_{\mathsf{d}}} - 1\right), & \hat{e}_i^{(j)} \leq \varepsilon_{\mathsf{d}}, \\ 1, & \varepsilon_{\mathsf{d}} < \hat{e}_i^{(j)} < \varepsilon_{\mathsf{r}}, \\ 1 + \frac{\hat{e}_i^{(j)} - \varepsilon_{\mathsf{r}}}{M^{(j)} - \varepsilon_{\mathsf{r}}}(\alpha_{\mathsf{r}} - 1), & \hat{e}_i^{(j)} \geq \varepsilon_{\mathsf{r}}, \end{cases}$$

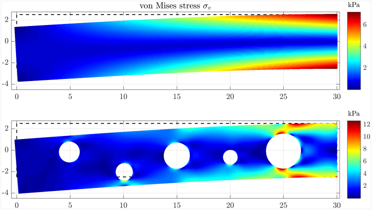
Refinement

Testing how the method behaves with variable density.

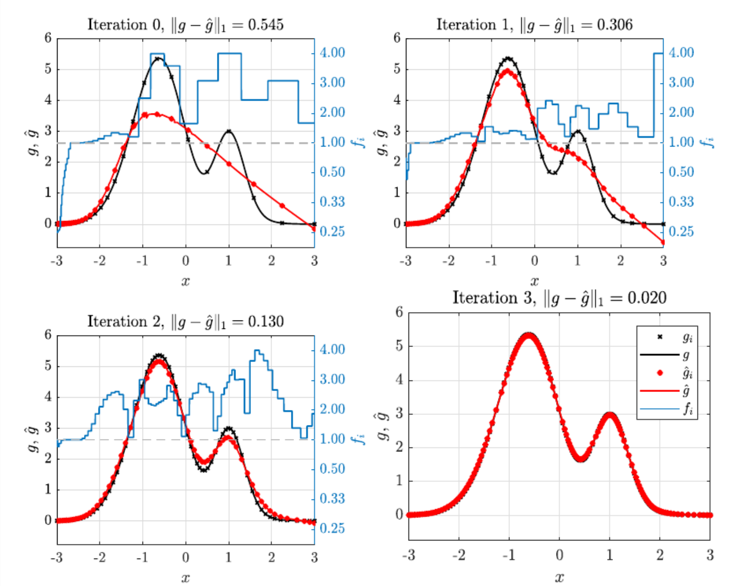


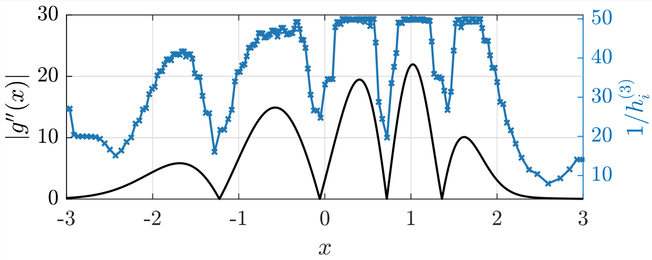






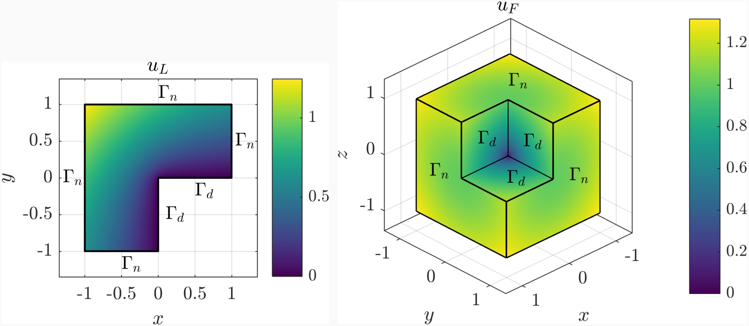
[SK19] Slak, Jure, and Gregor Kosec. "Refined Meshless Local Strong Form solution of Cauchy–Navier equation on an irregular domain." *Engineering analysis with boundary elements* 100 (2019): 3–13.


Iteration 0, $\|g - \hat{g}\|_1 = 0.545$
Iteration 1, $\|g - \hat{g}\|_1 = 0.306$
Iteration 2, $\|g - \hat{g}\|_1 = 0.130$
Iteration 3, $\|g - \hat{g}\|_1 = 0.020$
$g, \hat{g}$
f_i
x
4.00
3.00
2.00
1.00
0.50
0.33
0.25
g_i
g
$\hat{g}_i$
$\hat{g}$
f_i



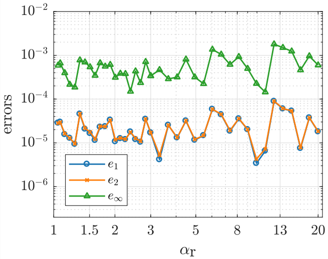
Spacing adapts to the error (proportional to 2nd derivative).

L-shaped domain and Fichera corner.

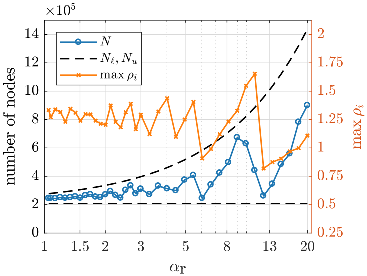


Not the cases where adaptivity truly shines, because we still get convergence with uniform refinement. Good for analysis.

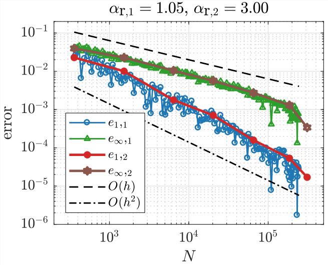
Analysis



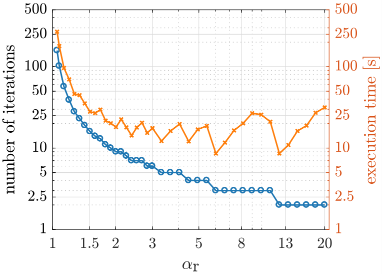
errors
e_1
e_2
e_∞
α_{r}



$\times 10^5$
number of nodes
N
N_ℓ, N_u
$\max \rho_i$
α_{r}

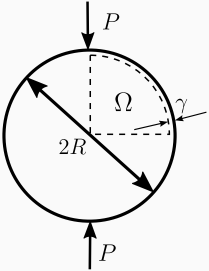


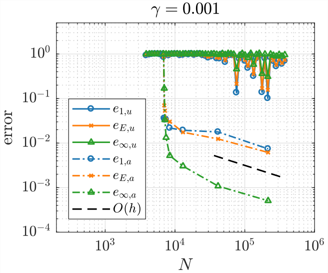
$\alpha_{\mathrm{r},1} = 1.05,\ \alpha_{\mathrm{r},2} = 3.00$
error
$e_{1,1}$
$e_{\infty,1}$
$e_{1,2}$
$e_{\infty,2}$
$O(h)$
$O(h^2)$
N

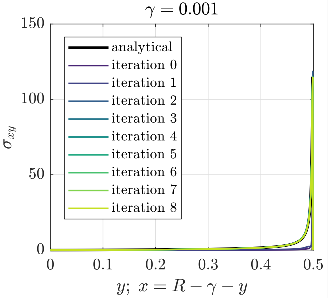


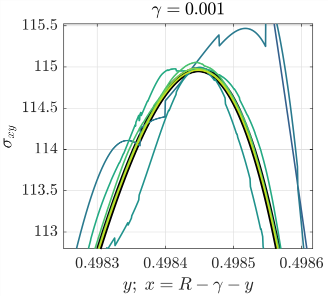
number of iterations
execution time [s]
α_{r}

Compressed disk

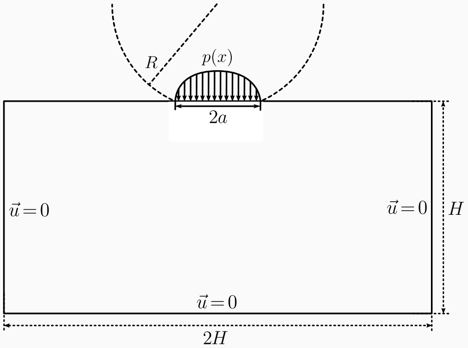








Hertzian contact

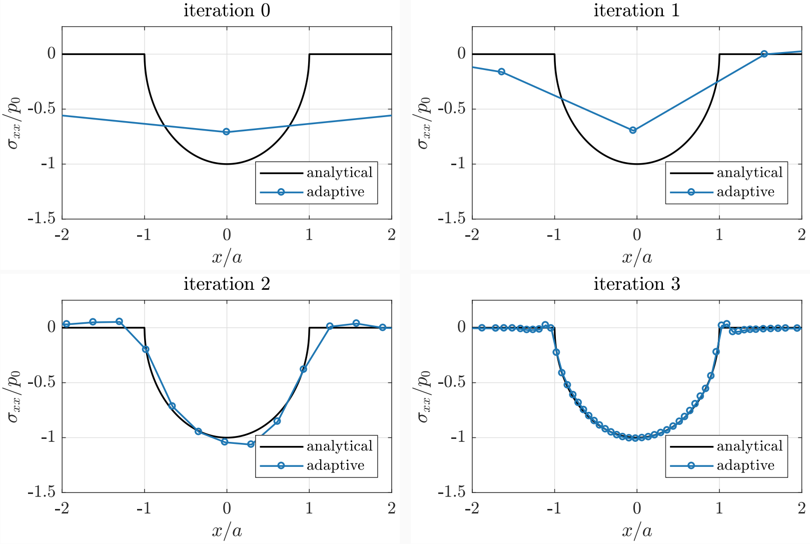


$H \approx 1923a$

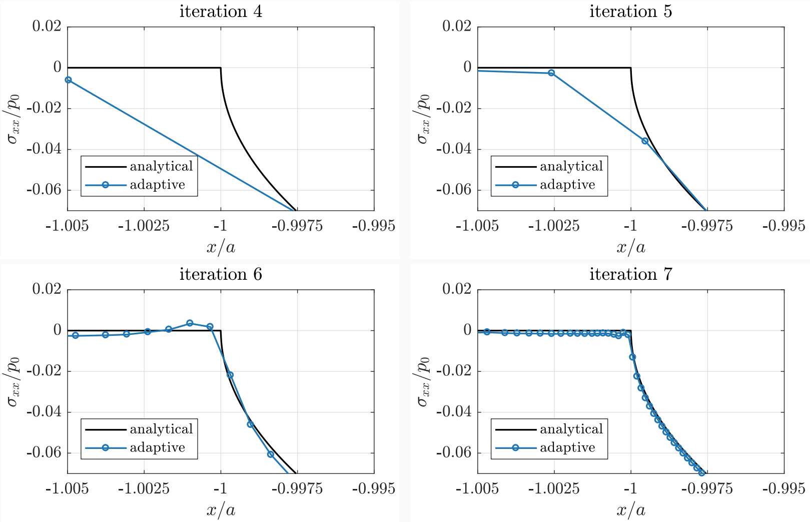
Closed form solution is complicated, but known.

Hertzian contact

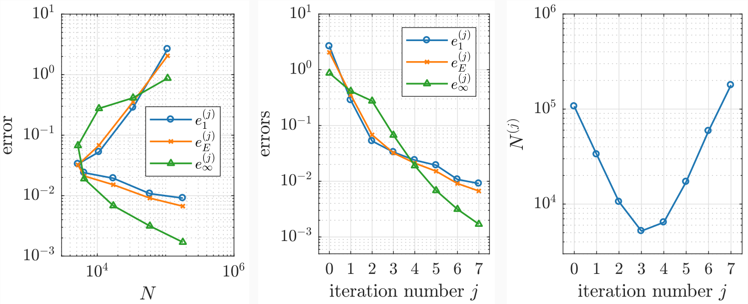
JSI



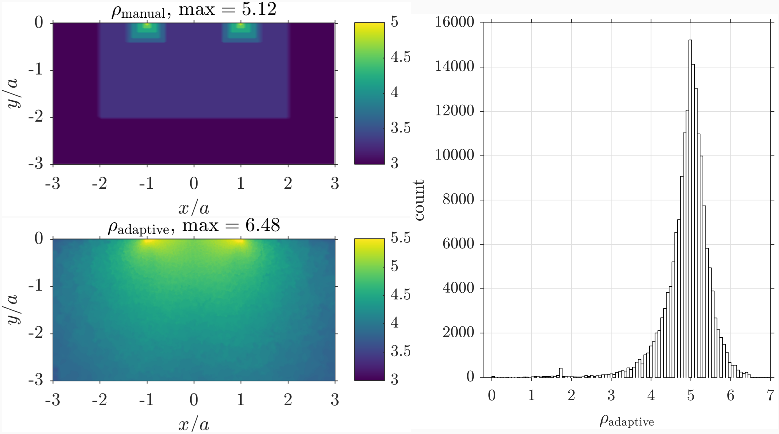
Hertzian contact



Hertzian contact

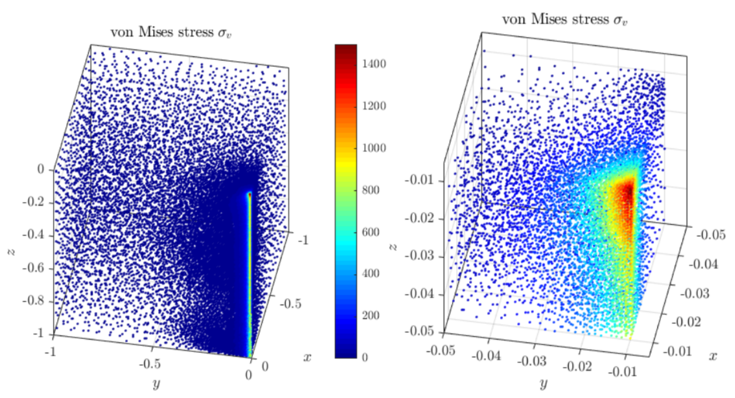


Derefinement works. Extreme refinement, distance ratio of 3 million.



Nodes with $\rho_i < 1$ cover 97% of the domain, but 95% of all nodes are in $[-3a, 3a] \times [-3a, 0]$ (which is 0.000027% of domain area).

3D case – point contact



Technical aspects

Result: an open source Medusa library for solving PDEs with strong-form methods. Submitted to TOMS, response: minor rev.

[SK20] J. Slak and G. Kosec, Medusa: A C++ library for solving PDEs using strong form mesh-free methods, arXiv:1912.13282

Medusa

Coordinate Free Meshless Method implementation

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

More details about the design and further examples in the thesis.

Future work

JSI

- Node generation from CAD models
- Parallel node generation
- Proofs on upper bound on $h_{X,\Omega}$
- Partial discretization modification
- Error indicators
- Approximation types RBF-FD vs. WLS
- Effect of stencil size
- Sensitivity to nodal positions (scattered uniform)
- Sensitivity to nodal positions (gradual density increase)
- h-based adaptivity instead of k-nn
- hp-adaptivity
- Time-dependent equations

Thank you for your attention. <jjNaturally, feel free to ask questions.