Fast and Robust Solvers for Domain Parameterizations within G+Smo

Delft University of Technology[1], Dalian University of Technology[2]

Ye Ji[1,2,*], Hugo M. Verhelst[1], Matthias Möller[1]

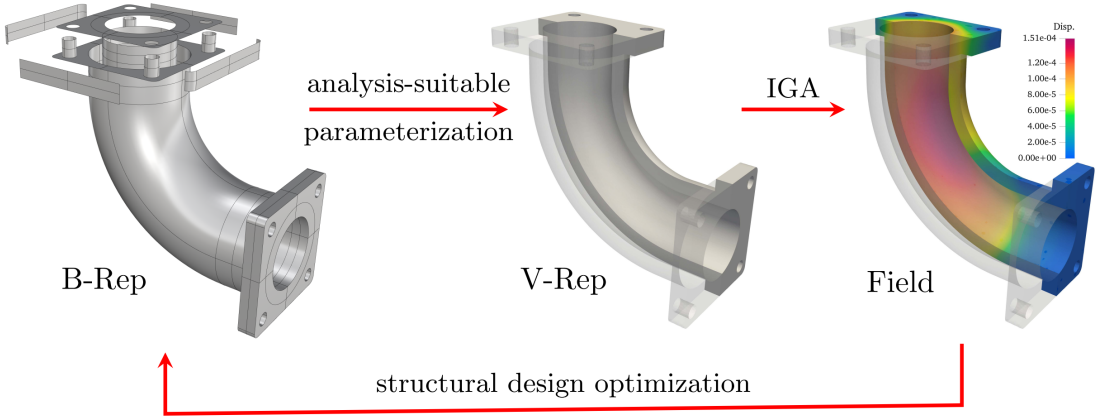
18–21 June 2023, Lyon (France), IGA 2023

Agenda

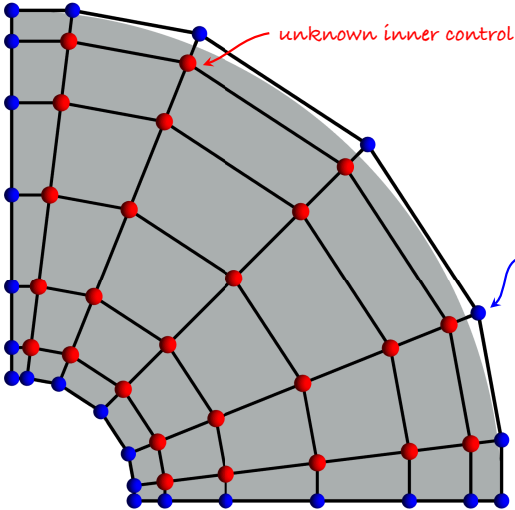
1. Research Background and Motivation
2. Nonlinear System Solvers
3. Nonlinear Optimization Solvers
4. Experiments and Results
5. Conclusions and Outlook

Domain Parameterization

- CAD models are usually represented by boundary representation (B-Rep).
- However, IGA requires an internal spline-based parameterization (V-Rep).



Problem Statement: Domain Parameterization



$$\boldsymbol{x}(\boldsymbol{\xi}) = \underbrace{\sum_{i \in \mathcal{I}_I} \mathbf{P}_i R_i(\boldsymbol{\xi})}_{\text{unknown}} + \underbrace{\sum_{j \in \mathcal{I}_B} \mathbf{P}_j R_j(\boldsymbol{\xi})}_{\text{known}}.$$

GOAL: To construct the **unknown inner control points** $\mathbf{P}_i$ (or basis functions $R_i(\boldsymbol{\xi})$) such that $\boldsymbol{x}$ ensures **bijectivity** and exhibits **minimal angle and area/volume distortion**.

Two Key Problem Categories in Domain Parameterization

- **PDE-based Methods:** Nonlinear vector-valued second-order PDE [1]

$$\begin{cases} \widetilde{\mathcal{L}}x = 0, \\ \widetilde{\mathcal{L}}y = 0, \end{cases} \quad \text{s.t. } \boldsymbol{x}|_{\partial\hat{\Omega}} = \partial\Omega.$$

- **Optimization-based Methods:** Minimization of objective function [2]

$$\begin{aligned} &\underset{\mathbf{P}_i,\ i\in\mathcal{I}_I}{\text{minimize}} && \mathscr{E}(\boldsymbol{x}) = \lambda_{\text{angle}}\mathscr{E}^{\text{angle}}(\boldsymbol{x}) + \lambda_{\text{vol}}\mathscr{E}^{\text{vol}}(\boldsymbol{x}) \\ &\text{s.t.} && \boldsymbol{x} \text{ is bijective.} \end{aligned}$$

- **Key Challenge:** Developing **Fast and Robust Solvers**.

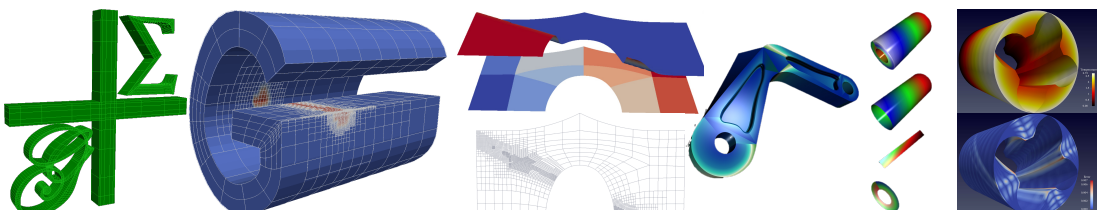
---

[1] Ref.: Hinz, J. P. (2020). PDE-Based Parameterization Techniques for Isogeometric Analysis Applications.
Ji, Y. et al. (2023). On an improved PDE-based elliptic ... Computer Aided Geometric Design, 102, 102191.

[2] Ref.: Ji, Y. et al. (2021). Constructing high-quality ... Journal of Computational and Applied Mathematics, 396, 113615.
Ji, Y. et al. (2022). Penalty function-based volumetric ... Computer Aided Geometric Design, 94, 102075.

Geometry Plus Simulation Module (G+Smo)

- Key features of G+Smo :
  - **Open-source, cross-platform, and template-based** C++ library for IGA.
  - Follows the principles of generic programming for **efficiency and ease of use**.
  - Provides **mathematical tools for geometric design and numerical simulation**.
  - Supports various bases: Bernstein, NURBS, (Truncated) Hierarchical B-splines.
  - Offers a **unified framework** for design and analysis pipelines.
  - Enables **dimension-independent** code through template meta-programming.



G+Smo Gallery, source

Agenda

1. Research Background and Motivation
2. Nonlinear System Solvers
3. Nonlinear Optimization Solvers
4. Experiments and Results
5. Conclusions and Outlook

Picard Iteration

Bratu problem (64×64):

$$\Delta u + \lambda e^u = 0, \quad in \quad \Omega,$$
$$u = 0, \quad on \quad \partial\Omega.$$

Nonlinear system $\iff$ Fixed-point iteration :

$$\mathcal{F}(\boldsymbol{u}) = 0 \iff \boldsymbol{u} = \mathcal{G}(\boldsymbol{u}) = \boldsymbol{u} + \mathcal{F}(\boldsymbol{u})$$

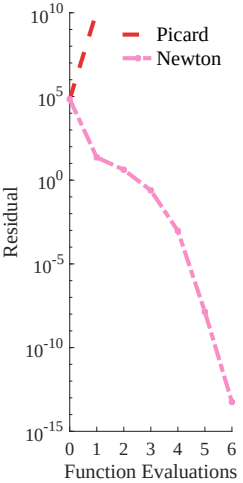
for some $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^n$.

- Basic Fixed-Point Iteration:

Algorithm Picard: Picard Iteration

1 Given $\boldsymbol{u}_0$.
2 **for** $k = 1, 2, \ldots, itmax$ *until* $\|\mathcal{F}_k\| < tol$ **do**
3 Set $\boldsymbol{u}_{k+1} = \mathcal{G}(\boldsymbol{u}^k)$

- **Converges too slowly to be useful, may even diverge.**



Anderson Acceleration (D.G. Anderson, 1965)

- AA(m) linearly recombines m **previous iterates** in a manner that approximately **minimizes the linearized fixed-point residual $\mathcal{F}$ in a least-squares fashion**.

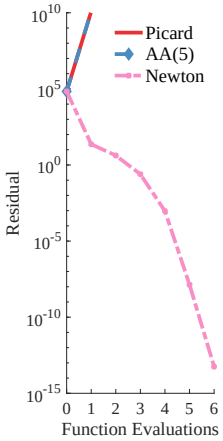
Algorithm AA: Anderson Acceleration

Given $\boldsymbol{u}_0$ and window size $m \geq 1$;
Set $\boldsymbol{u}_1 = \mathcal{G}(\boldsymbol{u}_0)$;
for $k = 1, 2, \ldots, itmax$ *until* $\|\mathcal{F}_k\| < tol$ **do**
  Set $m_k = \min\{m, k\}$;
  Determine $\boldsymbol{\alpha}^{(k)} = \left(\alpha_0^{(k)}, \alpha_1^{(k)}, \ldots, \alpha_{m_k}^{(k)}\right)^{\mathrm{T}}$ that solves

$$\underset{\boldsymbol{\alpha}=(\alpha_0,\alpha_1,\ldots,\alpha_{m_k})^{\mathrm{T}}}{\arg\min} \left\| \sum_{i=0}^{m_k} \mathcal{F}_{k-m_k+i}\boldsymbol{\alpha} \right\|_2^2, \quad \text{s.t.} \quad \sum_{i=0}^{m_k} \alpha_i = 1;$$

  Update $\boldsymbol{u}_{k+1} = \sum_{i=0}^{m_k} \alpha_i^{(k)} \mathcal{G}_{k-m_k+i}$;

Bratu problem (64×64):



Enhancing Anderson Acceleration with Preconditioning

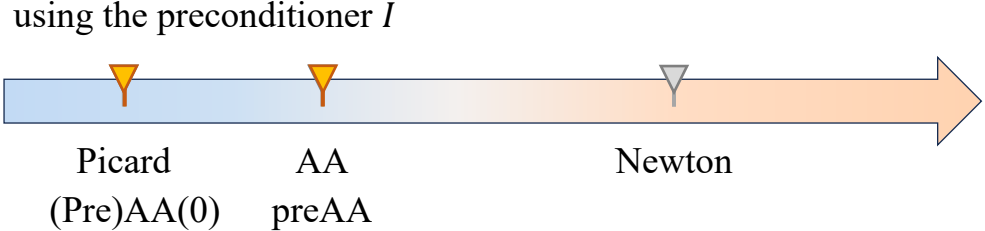
Preconditioning Strategy

We introduce a preconditioning strategy for the fixed-point iteration scheme:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \boldsymbol{\mathcal{M}}_k{}^{-1}\boldsymbol{\mathcal{F}}_k,$$

where $\boldsymbol{\mathcal{M}}_k$ is a non-singular matrix, known as the **preconditioner** at iteration k.

- By setting $\boldsymbol{\mathcal{M}}_k$ to the **identity matrix**, PreAA degenerates to the original AA.
- AA(0) is essentially the same as Picard iteration.



Enhancing Anderson Acceleration with Preconditioning

Preconditioning Strategy

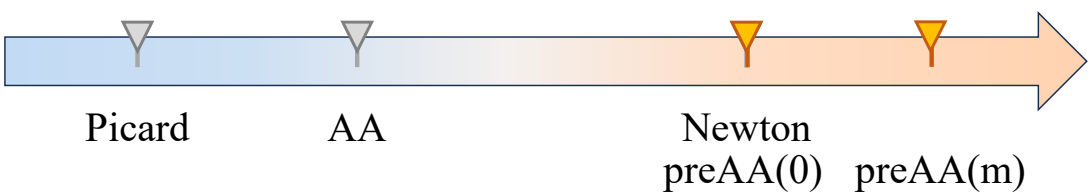
We introduce a preconditioning strategy for the fixed-point iteration scheme:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \boldsymbol{\mathcal{M}}_k{}^{-1}\boldsymbol{\mathcal{F}}_k,$$

where $\boldsymbol{\mathcal{M}}_k$ is a non-singular matrix, known as the **preconditioner** at iteration k.

- By setting $\boldsymbol{\mathcal{M}}_k$ to $jac(\boldsymbol{\mathcal{F}})$, PreAA(0) degenerates to Newton iteration.
- In this case, PreAA(m) leverages AA to accelerate Newton iteration.

using the preconditioner jac($\boldsymbol{\mathcal{F}}$)



Enhancing Anderson Acceleration with Preconditioning

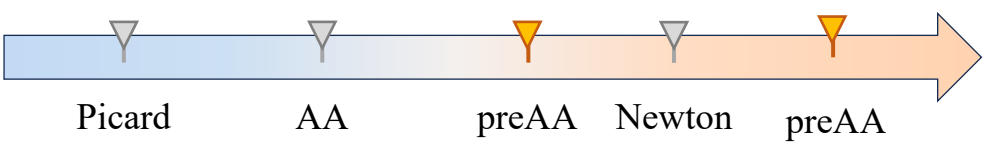
Preconditioning Strategy

We introduce a preconditioning strategy for the fixed-point iteration scheme:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \boldsymbol{\mathcal{M}}_k{}^{-1}\boldsymbol{\mathcal{F}}_k,$$

where $\boldsymbol{\mathcal{M}}_k$ is a non-singular matrix, known as the **preconditioner** at iteration k.

- **Ideal preconditioner:** Close to $jac(\boldsymbol{\mathcal{F}})$, yet computationally inexpensive.
- More flexibility, e.g., constant $\alpha\boldsymbol{I}$, diagonal Jacobian $diagJac(\boldsymbol{\mathcal{F}})$, upper (lower) triangular Jacobian $TriU(jac(\boldsymbol{\mathcal{F}}))$ and block-diagonal Jacobian $diagBlockJac(\boldsymbol{\mathcal{F}})$.



Enhancing Anderson Acceleration with Preconditioning

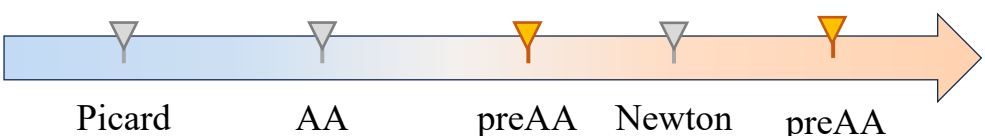
Preconditioning Strategy

We introduce a preconditioning strategy for the fixed-point iteration scheme:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \boldsymbol{\mathcal{M}}_k{}^{-1}\boldsymbol{\mathcal{F}}_k,$$

where $\boldsymbol{\mathcal{M}}_k$ is a non-singular matrix, known as the **preconditioner** at iteration k.

- **Ideal preconditioner:** Close to $jac(\boldsymbol{\mathcal{F}})$, yet computationally inexpensive.
- More flexibility, e.g., constant $\alpha\boldsymbol{I}$, diagonal Jacobian $diagJac(\boldsymbol{\mathcal{F}})$, upper (lower) triangular Jacobian $TriU(jac(\boldsymbol{\mathcal{F}}))$ and block-diagonal Jacobian $diagBlockJac(\boldsymbol{\mathcal{F}})$.
- **Delayed Update Strategy:** Reduces frequent preconditioner updates.



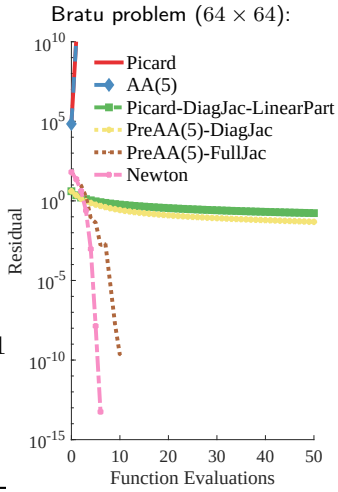
PreAA: Preconditioned Anderson Acceleration

Algorithm PreAA: preAndersonAccelerationpp

Given $\boldsymbol{u}_0$ and window size $m \geq 1$;
Set $\boldsymbol{u}_1 = \boldsymbol{\mathcal{G}}(\boldsymbol{u}_0)$;
for $k = 1, 2, \ldots, itmax$ *until* $\|\boldsymbol{\mathcal{F}}(\boldsymbol{u})\| < tol$ **do**
    // Update preconditioner
    if k *is evenly divisible by* N_{update} **then**
        Update preconditioning matrix $\boldsymbol{\mathcal{M}}_k$;
    Set $m_k = \min\{m, k\}$;
    Compute $\boldsymbol{\mathcal{E}}_k$ by solving $\boldsymbol{\mathcal{M}}_k \boldsymbol{\mathcal{E}}_k = -\boldsymbol{\mathcal{F}}_k$;
    Determine $\boldsymbol{\alpha}^{(k)} = \left(\alpha_0^{(k)}, \alpha_1^{(k)}, \ldots, \alpha_{m_k}^{(k)}\right)^{\mathrm{T}}$ that solves

$$\underset{\boldsymbol{\alpha}=(\alpha_0,\alpha_1,\ldots,\alpha_{m_k})^{\mathrm{T}}}{\arg\min} \left\| \sum_{i=0}^{m_k} \boldsymbol{\mathcal{E}}_{k-m_k+i} \boldsymbol{\alpha} \right\|_2^2, \quad \text{s.t.} \quad \sum_{i=0}^{m_k} \alpha_i = 1$$

    Update $\boldsymbol{u}_{k+1} = \sum_{i=0}^{m_k} \alpha_i^{(k)} \left(\boldsymbol{u}_{k-m_k+i} + \boldsymbol{\mathcal{E}}_{k-m_k+i}\right)$;



Convergence Theory

Theorem (**Residual bounds**)

Assume that the non-singular preconditioner $\boldsymbol{\mathcal{M}}_k$ satisfies $\|\mathbf{I} - \boldsymbol{\mathcal{M}}_k^{-1}\boldsymbol{\mathcal{J}}_k\| \leq L_k$, then the errors generate by preconditioned Anderson acceleration algorithm satisfy

$$\|e_{k+1}\| \leq \boldsymbol{\mathcal{C}} \sum_{j=0}^{m} \|e_{k-j}\|,$$

where $\boldsymbol{\mathcal{C}} = \max\{L_k, L_{k-1}, \cdots, L_{k-m}\} \cdot \max\{2\|\Gamma_k\|_\infty, 1 + \|\Gamma_k\|_\infty\}$ with $\Gamma_k = (\gamma_1, \gamma_2, \cdots, \gamma_m)^{\mathrm{T}}$.

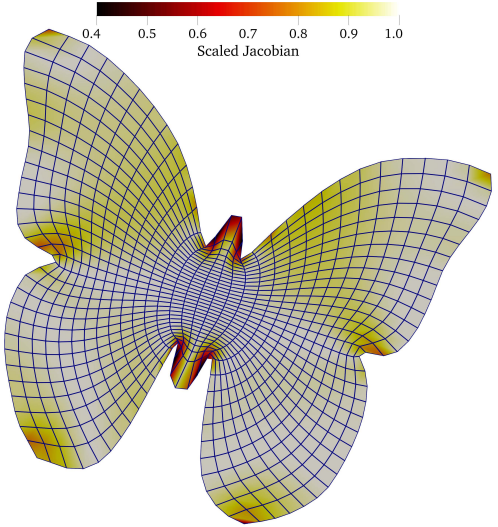
Corollary

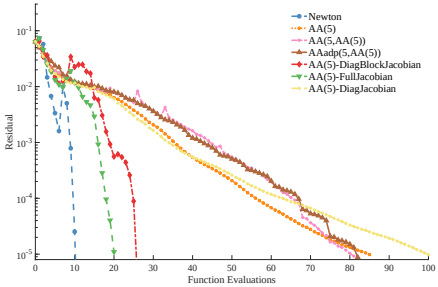
If we select a non-singular preconditioner $\boldsymbol{\mathcal{M}}_k$ that is sufficiently close to the Jacobian matrix $\boldsymbol{\mathcal{J}}_k$, such that

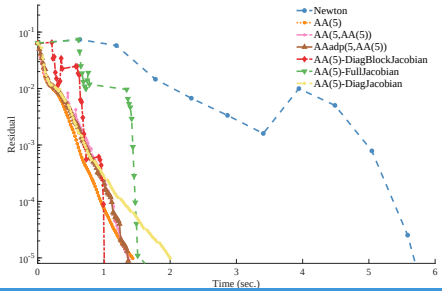
$$\left\|\mathbf{I} - \mathbf{M}_k^{-1}\boldsymbol{\mathcal{J}}_k\right\| \leq L_k < 1,$$

then the preconditioned Anderson acceleration $\mathbf{PreAA(m)}$ converges.

Butterfly Example: Performance Comparisons

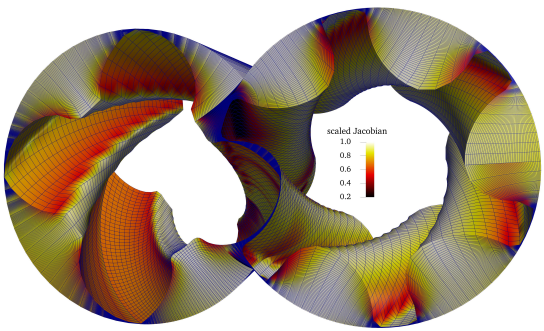


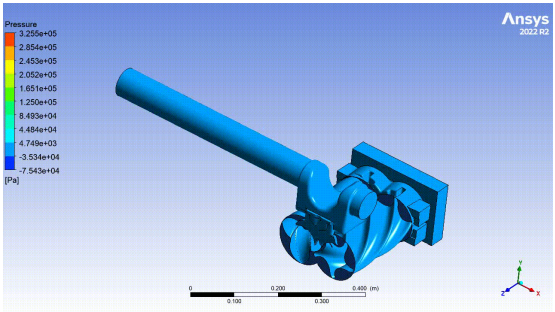




Rotary Twin-Screw Compressor Application

- PDE-based method: Excels in domains with extreme aspect ratios.
- **Real-world application:** Used in a rotary twin-screw compressor simulation.





Agenda

1. Research Background and Motivation
2. Nonlinear System Solvers
3. **Nonlinear Optimization Solvers**
4. Experiments and Results
5. Conclusions and Outlook

Limited-memory BFGS (L-BFGS) Algorithm

Algorithm L-BFGS: L-BFGS Algorithm

Given $\boldsymbol{u}_0$;
for $k = 1, 2, \ldots, itmax$ *until converged* **do**
  Compute gradient $\boldsymbol{g}_k$ at point $\boldsymbol{u}_k$;
  Line search for step size α_k;
  Update $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \alpha_k \cdot \boldsymbol{H}_k \cdot \boldsymbol{g}_k$;
  Compute gradient $\boldsymbol{g}_{k+1}$ at point $\boldsymbol{u}_{k+1}$;
  Compute difference $\boldsymbol{s}_k = \boldsymbol{u}_{k+1} - \boldsymbol{u}^k$;
  Compute gradient difference $\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$;
  $\boldsymbol{H}_{k+1} = \text{update\_Hessian\_inverse}(\boldsymbol{H}_k, \boldsymbol{s}_k, \boldsymbol{y}_k)$;

- A rank-2 correction quasi-Newton method.
- Considered one of the most stable and efficient unconstrained optimization solvers.
- Linear search criteria **ensure a sufficient function decrease** during the iteration.
- Open-source implementations available as external modules in G+Smo:
  HLBFGS: a hybrid L-BFGS routine, LBFGSpp: a header-only C++ library.

Objective Function Terms

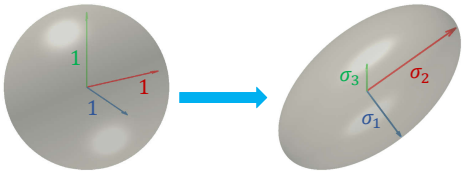
Angle distortion

- MIPS energy [HG2000, Fu+2015]:

$$\mathscr{E}^{\text{angle}} = \begin{cases} \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}, \\ \frac{1}{8}\left(\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}\right)\left(\frac{\sigma_2}{\sigma_3} + \frac{\sigma_3}{\sigma_2}\right)\left(\frac{\sigma_1}{\sigma_3} + \frac{\sigma_3}{\sigma_1}\right), \end{cases}$$

where σ_i are the singular values of $\mathcal{J}$.

- Ideally, $\sigma_1 = \sigma_2 = \cdots = \sigma_d$.

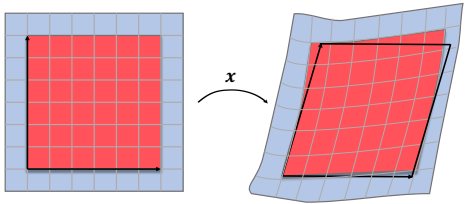


Area/Volume distortion

- Area/Volume distortion energy:

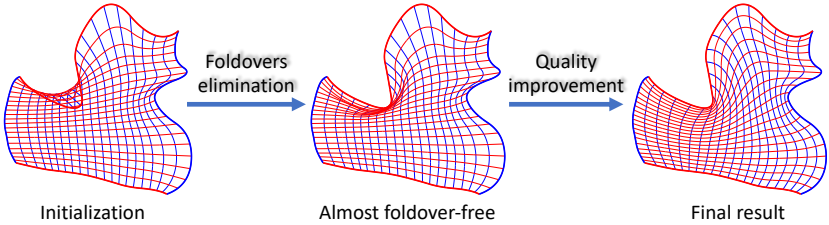
$$\mathscr{E}^{\text{unif.}}(\boldsymbol{x}) = \frac{|\mathcal{J}|}{vol(\Omega)} + \frac{vol(\Omega)}{|\mathcal{J}|},$$

where $vol(\Omega)$ denotes the area/volume of the computational domain Ω;



Optimization-based Parameterization Approaches

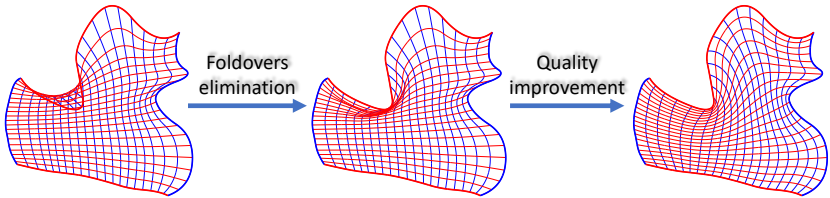
- Barrier function-based method [1]:



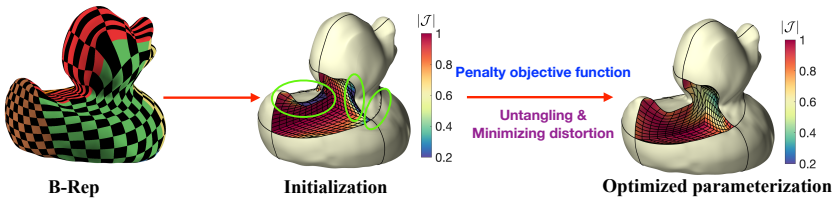
[1] Ji, Y. et al. (2021). Constructing high-quality ... Journal of Computational and Applied Mathematics, 396, 113615.

Optimization-based Parameterization Approaches

- Barrier function-based method [1]:



- Penalty function-based method [2]:

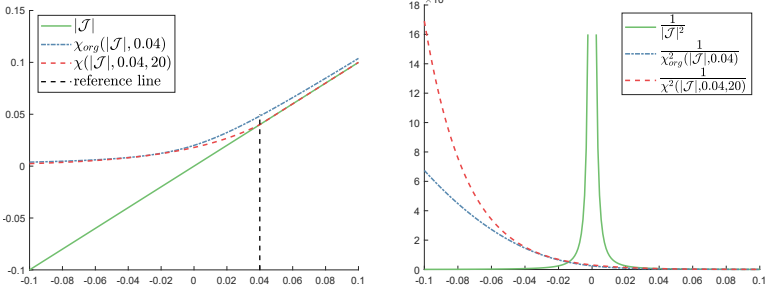


[1] Ji, Y. et al. (2021). Constructing high-quality ... Journal of Computational and Applied Mathematics, 396, 113615.

[2] Ji, Y. et al. (2022). Penalty function-based volumetric ... Computer Aided Geometric Design, 94, 102075.

Why it Works?

- Penalty functions χ:

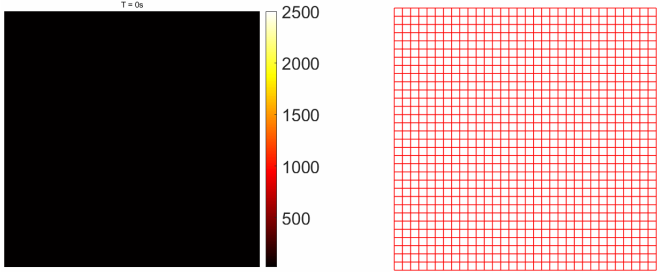


- Key idea: The line search criteria **guarantee a sufficient function decrease!!**

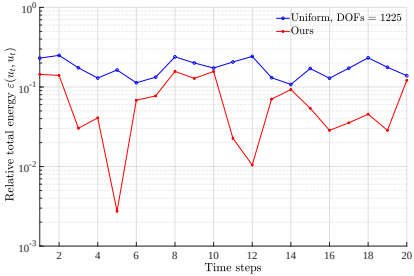
Application: r-Adaptive Parameterization [3]

- 2D **linear heat transfer problem** with a moving Gaussian heat source:

$$\begin{cases} C_p\rho\frac{\partial u(\boldsymbol{x},t)}{\partial t} - \nabla\cdot(\kappa\nabla u(\boldsymbol{u},t)) = f(\boldsymbol{x},t) & \text{in } \Omega\times T \\ u(\boldsymbol{x},t) = u_0 & \text{in } \Omega \\ \kappa\nabla u(\boldsymbol{x},t) = 0 & \text{on } \partial\Omega\times T \end{cases}$$



$u(\boldsymbol{x},t)$ and the parameterizations



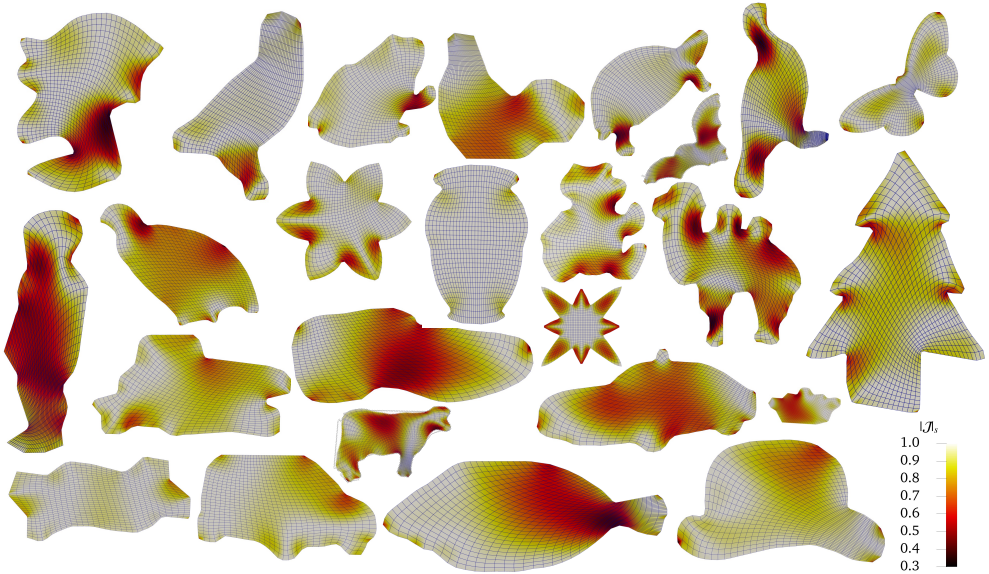
Errors vs. time instants t

[3] Ji Y. et al. (2022) Curvature-based R-Adaptive ... Computer-Aided Design, 150: 103305.

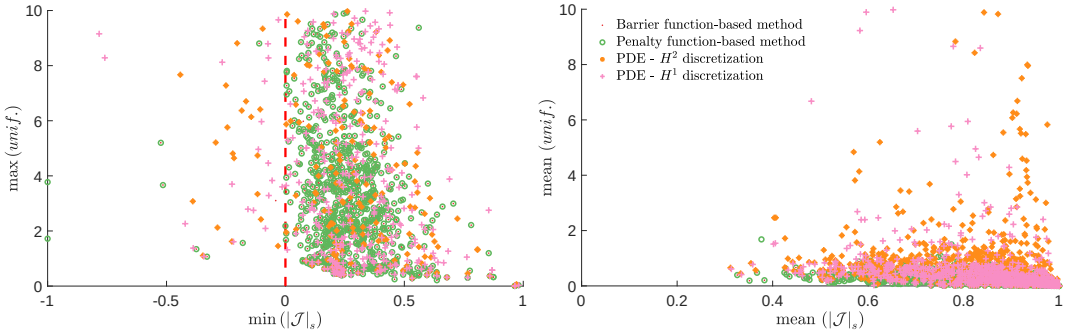
Agenda

1. Research Background and Motivation
2. Nonlinear System Solvers
3. Nonlinear Optimization Solvers
4. **Experiments and Results**
5. Conclusions and Outlook

Planar Parameterization Test Dataset (977 models)



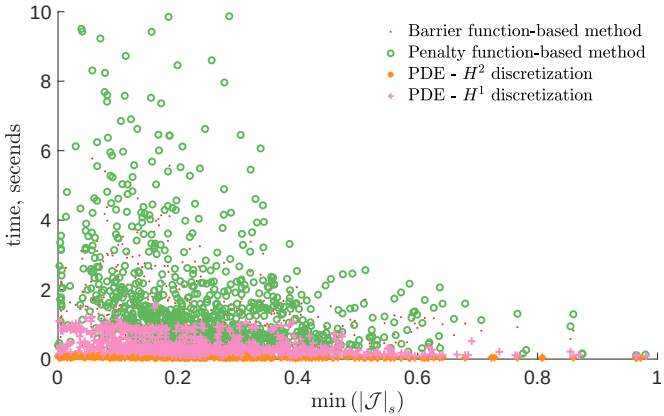
Effectiveness and Quality Assessment



Success rates:

- PDE - H^2 discretization: $608/977 \simeq 62.23\%$;
- PDE - H^1 discretization: $721/977 \simeq 73.80\%$;
- Barrier function-based method: $961/977 \simeq$ **98.36%**;
- Penalty function-based method: $956/977 \simeq 97.85\%$.

Computational Time



- PDE-based $\sim$ 0.2 sec., optimization based $\sim$ 2 sec. on my laptop. (DOFs=1250)
- PDE-based methods demonstrate higher efficiency.

Agenda

1. Research Background and Motivation
2. Nonlinear System Solvers
3. Nonlinear Optimization Solvers
4. Experiments and Results
5. Conclusions and Outlook

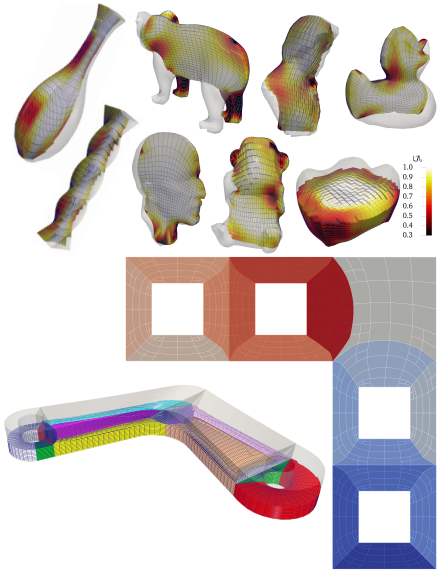
Conclusions and Outlook

Conclusions:

- **PreAA:** Proposed for PDE-based methods.
- **L-BFGS:** Utilized in optimization-based methods and r-adaptive approach.

Future Work:

- **Topology Computation:** Investigate multi-patch parameterization.
- **Adaptive Methods:** Enhance efficiency.



Thanks for your attention!

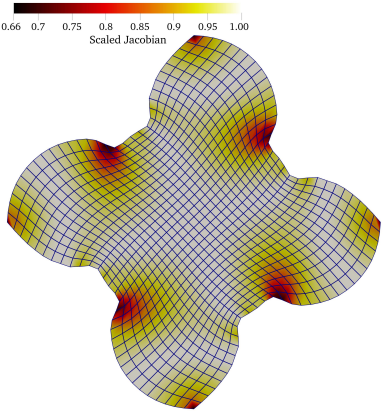
Q&A.

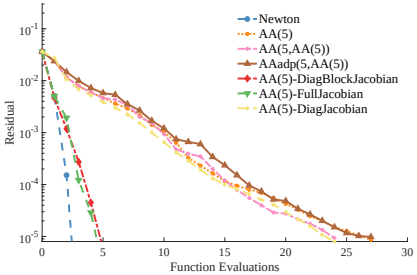
If interested in my research, please feel free to contact me! ;-)

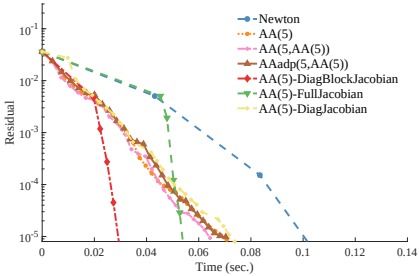
- email: jiyess@outlook.com
- homepage: https://jiyess.github.io
- github: jiyess

I am looking for a postdoctoral position!

Male rotor example: performance comparisons







Equivalence Problem: Unconstrained Optimization

- Recall the planar MIPS energy,

$$\begin{aligned}\mathscr{E}_{2D}^{\text{angle}}(\boldsymbol{x}) &= \frac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2} \\ &= \frac{trace(\boldsymbol{\mathcal{J}}^{\mathrm{T}}\boldsymbol{\mathcal{J}})}{|\boldsymbol{\mathcal{J}}|}.\end{aligned}$$

  Since the Jacobian determinant appears in its denominator, it proceeds to infinity if the Jacobian determinant $|\boldsymbol{\mathcal{J}}|$ approaches zero.

- Solve the following **unconstrained optimization problem**:

$$\underset{\mathbf{P}_i,\ i\in\mathcal{I}_I}{\arg\min}\quad \mathscr{E}(\boldsymbol{x}) = \int_{\hat{\Omega}} \left(\lambda_{\text{angle}}\mathscr{E}^{\text{angle}}(\boldsymbol{x}) \ + \lambda_{\text{vol}}\mathscr{E}^{\text{vol}}(\boldsymbol{x})\right)\ \mathrm{d}\hat{\Omega}. \tag{1}$$

Local/Global Parameterization