Implementation of analysis-suitable parameterization construction using G+Smo

Ye Ji

Delft University of Technology, The Netherlands
Dalian University of Technology, China

October 13, 2022
• Ye Ji et al., Constructing high-quality planar NURBS parameterization for isogeometric analysis by adjustment control points and weights, Journal of Computational and Applied Mathematics, 396 (2021), 113615.
IsoGeometric Analysis (IGA)

- Proposed by T.J.R. Hughes et al., 2005.
- **KEY IDEA:** approximate the physical fields with the same basis functions as that used to generate CAD models.
- Advantages:
  - Integration of design and analysis;
  - Exact and efficient geometry;
  - No data type transition and mesh generation;
  - Simplified mesh refinement;
  - High order continuous field;
  - Superior approximation properties.
- Very broad applications: such as shell analysis, fluid-structure interaction, and shape and topology optimization.

Source: Figure from [Cottrell et al. 2009]
Most modern CAD systems only focus on boundary representations (B-Reps) in geometry modeling.
Most modern CAD systems only focus on boundary representations (B-Reps) in geometry modeling.

**Problem statement:**
- From a given B-Rep, constructing an analysis-suitable parameterization $x$. 

Most modern CAD systems only focus on boundary representations (B-Reps) in geometry modeling.

**Problem statement:**
- From a given B-Rep, constructing an analysis-suitable parameterization $x$.
- Analysis-suitable parameterizations should
  - be **bijective**;
  - ensure as **low angle and volume distortion** as possible.
Problem statement

- A spline-based parameterization $\mathbf{x}$ from a parametric domain $\mathcal{P} = [0, 1]^d$ ($d = 2, 3$) to computational domain $\Omega$ is of the following form

$$\mathbf{x}(\xi) = \mathbf{R}^T \mathbf{P} = \sum_{i \in \mathcal{I}_I} \mathbf{P}_i \mathbf{R}_i(\xi) + \sum_{j \in \mathcal{I}_B} \mathbf{P}_j \mathbf{R}_j(\xi),$$  \hspace{1cm} (1)

where $\mathbf{P}_i$ are unknown inner control points and $\mathbf{P}_j$ are given boundary control points.

- **GOAL:** To construct the unknown inner control points $\mathbf{P}_i$ such that $\mathbf{x}$ is bijective and has the lowest possible angle and area/volume distortion.
Objective function: angle distortion

- Most-Isometric ParameterizationS (MIPS) energy [Hormann and Greiner 2000, Fu et al. 2015]:

\[
E_{\text{angle}}(\mathbf{x}) = \begin{cases} 
\frac{1}{8} \left( \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} \right) \left( \frac{\sigma_2}{\sigma_3} + \frac{\sigma_3}{\sigma_2} \right) \left( \frac{\sigma_3}{\sigma_1} + \frac{\sigma_1}{\sigma_3} \right), & 2D, \\
\frac{1}{8} \left( \sigma_1 \sigma_2 + \sigma_2 \sigma_1 \right), & 3D.
\end{cases}
\]

(2)

where \(\sigma_i\) are the singular values of the Jacobian matrix \(\mathbf{J}\) of the parameterization \(\mathbf{x}\).

- When \(\sigma_1 = \sigma_2 = \ldots = \sigma_d\), \(\mathbf{x}\) is conformal and \(E_{\text{angle}}\) reaches its minimum value.
Objective function: area/volume distortion

- Area/volume distortion energy:

\[ E_{\text{vol}}(x) = \frac{|\mathcal{J}|}{\text{vol}(\Omega)} + \frac{\text{vol}(\Omega)}{|\mathcal{J}|}, \]  

where \( \text{vol}(\Omega) \) denotes the area/volume of the computational domain \( \Omega \);
Objective function: variational formulation

**Basic idea:** to solve the following constrained optimization problem:

$$\arg \min_{\mathbf{P}_i, \ i \in I} E(\mathbf{x}) = \int_{\mathcal{P}} (\lambda_1 E_{\text{angle}}(\mathbf{x}) + \lambda_2 E_{\text{vol}}(\mathbf{x})) \ d\mathcal{P},$$

s.t. \( \mathbf{x} \) is bijective. 

Suppose that the given B-Rep is bijective. \( \mathbf{x} \) is bijective \( \iff \left| J(\mathbf{x}(\xi)) \right| \neq 0, \forall \xi \in \mathcal{P} \).

Due to the high-order continuity of \( \mathbf{x} \), we need \( \left| J \right| > 0 \) (\( < 0 \)), \( \forall \xi \in \mathcal{P} \).
Objective function: variational formulation

- **Basic idea:** to solve the following constrained optimization problem:

  $\arg\min_{\mathbf{P}_i, \ i \in \mathcal{I}} \ E(\mathbf{x}) = \int_{\mathcal{P}} \left( \lambda_1 E_{\text{angle}}(\mathbf{x}) + \lambda_2 E_{\text{vol}}(\mathbf{x}) \right) \ d\mathcal{P},$  
  
  \hspace{1cm} s.t. \ x \text{ is bijective.} \tag{4}$

- Suppose that the given B-Rep is bijective. $\mathbf{x}$ is bijective $\iff |\mathcal{J}(\mathbf{x}(\xi))| \neq 0, \ \forall \xi \in \mathcal{P}.$
Objective function: variational formulation

- **Basic idea**: to solve the following constrained optimization problem:

  $$\arg \min_{P_i, i \in I} E(x) = \int_{\mathcal{P}} \left( \lambda_1 E_{\text{angle}}(x) + \lambda_2 E_{\text{vol}}(x) \right) \, d\mathcal{P},$$

  \[ s.t. \quad x \text{ is bijective.} \]  

- Suppose that the given B-Rep is bijective. $x$ is bijective $\iff |\mathcal{J}(x(\xi))| \neq 0$, $\forall \xi \in \mathcal{P}$.
- Due to the high-order continuity of $x$, we need $|\mathcal{J}| > 0$ ($< 0$), $\forall \xi \in \mathcal{P}$. 
Treatment of bijectivity constraint

- The Jacobian determinant can be represented by a linear combination of splines

\[ |J| = \sum_i |J_i| N_i(\xi) \]  \hspace{1cm} (5)
Treatment of bijectivity constraint

- The Jacobian determinant can be represented by a linear combination of splines

\[
|\mathcal{J}| = \sum_i |\mathcal{J}_i| N_i(\xi)
\]  

(5)

- Many works handle the bijectivity constraint with inequality constraints $|\mathcal{J}_i| > 0$. [Xu et al., CMAME 2011, Wang and Qian 2014]
Treatment of bijectivity constraint

- The Jacobian determinant can be represented by a linear combination of splines
  \[ |\mathcal{J}| = \sum_i |\mathcal{J}_i| N_i(\xi) \]  \hspace{1cm} (5)

- Many works handle the bijectivity constraint with inequality constraints \(|\mathcal{J}_i| > 0\). [Xu et al., CMAME 2011, Wang and Qian 2014]

- However, the number of the constraints can be huge. [Pan et al., CMAME 2020, Ji et al. JCAM 2021]. (To a bi-cubic planar NURBS parameterization with 20 × 20 control points, the number of inequality constraints is over 34k.)
Equivalence problem: unconstrained optimization

- Recall the planar MIPS energy,

\[
E_{\text{angle}}^{2D}(x) = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1 \sigma_2} = \frac{\text{trace}(J^T J)}{|J|}.
\]

Since the Jacobian determinant appears in its denominator, it proceeds to infinity if the Jacobian determinant $|J|$ approaches zero.
Equivalence problem: unconstrained optimization

- Recall the planar MIPS energy,

\[ E_{angle}^{2D}(x) = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1 \sigma_2} = \frac{\text{trace}(\mathcal{J}^T \mathcal{J})}{|\mathcal{J}|}. \]

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

- Remove the constraints and solve the following unconstrained optimization problem:

\[
\arg \min_{\mathcal{P}_i, \ i \in I} ~ E(x) = \int_{\mathcal{P}} \left( \lambda_1 E_{angle}(x) + \lambda_2 E_{vol}(x) \right) \, d\mathcal{P}.
\]
Hybrid L-BFGS (HLBFGS) solver (NEW in G+Smo!)

(https://xueyuhanlang.github.io/software/HLBFGS/)

- A framework for unconstrained optimization problems written by Yang Liu (Microsoft Research Asia).
  - Light-weight and freely available for non-commercial purposes;
  - Unifies common optimization methods, such as gradient-decent method, (Preconditioned) L-BFGS method, (Preconditioned) Conjugate Gradient method, and Newton's method.
  - Very popular in computer graphics community.
Hybrid L-BFGS (HLBFGS) solver (NEW in G+Smo!)

(https://xueyuhanlang.github.io/software/HLBFGS/)

- A framework for unconstrained optimization problems written by Yang Liu (Microsoft Research Asia).
  - Light-weight and freely available for non-commercial purposes;
  - Unifies common optimization methods, such as gradient-decent method, (Preconditioned) L-BFGS method, (Preconditioned) Conjugate Gradient method, and Newton's method.
  - Very popular in computer graphics community.

- Already integrated into G+Smo (stable):
  - Example: examples/optimizer_example.cpp;
  - G+Smo wrapper: /extensions/gsHLBFGS/gsHLBFGS.h;
  - Source codes: /external/HLBFGS.
Basic usage of HLBFGS solver (NEW in G+Smo!)

```cpp
template <typename T>
class gsOptProblemExample : public gsOptProblem<T> {
public:
    // The constructor defines all properties of our optimization problem
    gsOptProblemExample() {};

    // The evaluation of the objective function must be implemented
    T evalObj(const gsAsConstVector<T> &u) const {};

    // The gradient (resorts to finite differences if unimplemented)
    void gradObj_into(const gsAsConstVector<T> &u, gsAsVector<T> &result) const {};
    ...
};
```
Basic usage of HLBFGS solver (NEW in G+Smo!)

```cpp
int main(int argc, char* argv[]){
    ...
    gsOptProblemExample<real_t> problem;
    gsHLBFGS<real_t> *optimizer;

    // Set stopping criterion for iteration (optional)
    optimizer->options().setInt("MaxIterations", 200);
    optimizer->options().setInt(...);

    // Set initial guess (optional)
    gsVector<real_t> initialGuess;
    initialGuess << ...;

    // Solve
    optimizer->solve(initialGuess);
    ...
}
```
Initialization

- Many algebraic methods can be adopted to initialize:
  - Discrete Coon’s patch [Farin and Hansford 1999];
  - Spring patch [Gravesen et al. 2012];
  - Smoothness energy minimization [Wang et al. 2003, Pan et al. 2020];
  - ...

- No guarantee of bijectivity.

- However, an already bijective parameterization is needed in our optimization problem (6).
Barrier function-based parameterization construction

- Three-step strategy.

Initialization → Foldovers elimination → Almost foldover-free → Quality improvement → Final result
Foldovers elimination: almost foldover-free parameterization

- Some works solve the following Max-Min problem:

\[
\arg \min_{P_i, i \in I} \max_j |J|_j,
\]

where \(|J|_j\) are the expansion coefficients of \(|J|\).

- High computational costs still but NOT necessary!
Foldovers elimination: almost foldover-free parameterization

- Some works solve the following Max-Min problem:

\[
\arg \min_{\mathbf{P}_i, \ i \in \mathcal{I}} \max_j |\mathcal{J}|_j,
\]

where \(|\mathcal{J}|_j\) are the expansion coefficients of \(|\mathcal{J}|\).

- **High computational costs still but NOT necessary!**

- We solve the following problem instead:

\[
\arg \min_{\mathbf{P}_i, \ i \in \mathcal{I}} E(\mathbf{x}) = \int_P \max (0, \delta - |\mathcal{J}|) \ d\mathbf{P},
\]

where \(\delta\) is a threshold (\(\delta = 5\% vol(\Omega)\) as default).
Foldovers elimination: almost foldover-free parameterization

```cpp
template<short_t d, typename T>
T gsObjFoldoverFree<d, T>::evalObj(const gsAsConstVector<T> &u) const
{
    // update m_mp with the current design
    convert_gsFreeVec_to_mp<T>(u, m_mapper, m_mp);
    geometryMap G = m_evaluator.getMap(m_mp);

    // defines the expression of integrand
    auto EfoldoverFree = (m_delta - jac(G).det()).ppartval();
    return m_evaluator.integral(EfoldoverFree);
}
```
Foldovers elimination: almost foldover-free parameterization

- Foldovers elimination is of vital importance. If it fails, everything CRASHES!!!
- For practical purposes, we gradually reduce the $\delta$.

```cpp
...  
gsObjFoldoverFree<d, T> objFoldoverFree(mp, mapper);
  do {
    T delta = pow(0.1, it) * 5e-2 * scaledArea; // update the parameter delta
    objFoldoverFree.options().setReal("ff_Delta", delta);
    objFoldoverFree.applyOptions();

    gsHLBFGS<T> optFoldoverFree(&objFoldoverFree);
    optFoldoverFree.solve(initialGuessVector); // solve the current problem

    Efoldover = optFoldoverFree.currentObjValue();
    initialGuessVector = optFoldoverFree.currentDesign();
    ++it;
  } while (Efoldover > 1e-20 && it < 10);
```
Quality improvement: robustness consideration

- Recall that $E_{\text{angle}}$ proceeds to infinity if the Jacobian determinant $\mathcal{J}$ approaches zero.
- DANGER!: discontinuous function value change in numerical optimization.

\[ E_{\text{angle}} = \begin{cases} \int P(\lambda_1 E_{\text{angle}}(x) + \lambda_2 E_{\text{vol}}(x)) \, dP, & \text{if } \min \mid \mathcal{J} \mid > 0, \\ +\infty, & \text{otherwise} \end{cases} \]
Quality improvement: robustness consideration

- Recall that $E_{\text{angle}}$ proceeds to infinity if the Jacobian determinant $J$ approaches zero.
- DANGER!: discontinuous function value change in numerical optimization.
- Line search ensures sufficient reduction, e.g., strong Wolfe condition.
Quality improvement: robustness consideration

- Recall that $E_{\text{angle}}$ proceeds to infinity if the Jacobian determinant $J$ approaches zero.
- **DANGER!**: discontinuous function value change in numerical optimization.
- Line search ensures sufficient reduction, e.g., strong Wolfe condition.
- With this feature, we simply revise the objective function (barrier function):

\[
E^c = \begin{cases} 
\int_{\mathcal{P}} \left( \lambda_1 E_{\text{angle}}(x) + \lambda_2 E_{\text{vol}}(x) \right) \, d\mathcal{P}, & \text{if } \min |J| > 0, \\
+\infty, & \text{otherwise.}
\end{cases}
\]
Quality improvement

```cpp
// Quality improvement

template<short_t d, typename T>
template<short_t _d>
typename std::enable_if<_d == 2, T>::type
gsObjQualityImprovePt<d, T>::evalObj(const gsAsConstVector<T> &u) const
{
    // update m_mp with the current design
    ...

    // set the objective function value to +\infty if min(jac(G).det()) < 0
    if (m_evaluator.min(jac(G).det()) < 0){return std::numeric_limits<T>::max();}
    else {
        // otherwise, compute the normal objective function value
        auto Euniform = chi / area + area / chi;
        auto Ewinslow = jac(G).sqNorm() / jac(G).det();
        return m_evaluator.integral(m_lambda1 * Ewinslow + m_lambda2 * Euniform);
    }
}  
```
Analytical gradient: for stability aspect

- In the class `gsOptProblem<T>`, we have a default `gradObj_into()` which approximate the gradient by numerical differentiation

\[
f'(x) = \frac{-f(x + 2h) + 8f(x + h) - 8f(x - h) + f(x + 2h)}{12h} + \frac{h^4}{30} f^{(5)}(c),
\]

where \( c \in [x - 2h, x + 2h] \).
Analytical gradient: for stability aspect

- In the class gsOptProblem<T>, we have a default gradObj_into() which approximate the gradient by numerical differentiation

\[
f'(x) = \frac{-f(x + 2h) + 8f(x + h) - 8f(x - h) + f(x + 2h)}{12h} + \frac{h^4}{30} f^{(5)}(c),
\]

where \( c \in [x - 2h, x + 2h] \).

- Hard to select a suitable step size \( h \), specifically for our problem.
Analytical gradient: for efficiency aspect

- To a single-patch tri-cubic B-spline parameterization with 25 control points along each direction (using standard Gauss quadrature rule), $4 \times 23^3 \times (3 + 1)^3 > 3$ M function evaluations are performed for once line-search.
Gallery: barrier function-based method
Penalty function-based parameterization construction

- The foldovers elimination does not improve sufficient to the parameterization quality.
Penalty function-based parameterization construction

- The foldovers elimination does not improve sufficient to the parameterization quality.
- Avoids extra foldovers elimination steps.
- **Untangling and minimizing distortion perform simultaneously!!!**

![Diagram showing the process from B-Rep to Optimized parameterization](image)
Basic idea: Penalty function

- **Penalty function:**

\[
\chi(|\mathcal{J}|, \varepsilon, \beta) = \begin{cases} 
\varepsilon \cdot e^{\beta(|\mathcal{J}| - \varepsilon)} & \text{if } |\mathcal{J}| \leq \varepsilon \\
|\mathcal{J}| & \text{if } |\mathcal{J}| > \varepsilon
\end{cases}
\]  

(7)

where \(\varepsilon\) is a small positive number and \(\beta\) is a penalty factor;

- \(\chi(|\mathcal{J}|, \varepsilon, \beta)\) equals a small positive number if \(|\mathcal{J}| < \varepsilon\), and strictly equals the Jacobian determinant \(|\mathcal{J}|\) if \(|\mathcal{J}| \geq \varepsilon\);

- \(\frac{1}{\chi^2(|\mathcal{J}|, \varepsilon, \beta)}\) have very large values to penalize the negative Jacobians and small values to accept positive Jacobians.
Jacobian regularization and revised objective function

- With this basic idea, we solve the following optimization problem:

\[
\arg \min_{\mathbf{P}_i, \ i \in \mathcal{I}} \mathcal{E}_c = \int_{\mathcal{P}} \left( \lambda_1 \mathcal{E}_{c \text{ mips}} + \lambda_2 \mathcal{E}_{c \text{ vol}} \right) \, d\mathcal{P}
\]

\[
= \int_{\mathcal{P}} \left( \frac{\lambda_1}{8} \kappa_F^2(\mathcal{J}) \cdot \frac{\left| \mathcal{J} \right|^2}{\chi^2(\left| \mathcal{J} \right|, \varepsilon, \beta)} + \lambda_2 \left( \frac{\text{vol}(\Omega)}{\chi(\left| \mathcal{J} \right|, \varepsilon, \beta)} + \frac{\chi(\left| \mathcal{J} \right|, \varepsilon, \beta)}{\text{vol}(\Omega)} \right) \right) \, d\mathcal{P},
\]

where \( \mathbf{P}_i, \ i \in \mathcal{I} \) are the unknown inner control points.

- Now, **only one optimization problem is solved**.
Gallery: penalty function-based results
Timing comparisons of G+Smo and MATLAB implementations

- G+Smo code is not as fast as we expected.
Timing comparisons of G+Smo and MATLAB implementations

- G+Smo code is not as fast as we expected.
- Precompute basis functions (unsteady problems and structural optimization).
G+Smo implementation with OPENMP

Timings (sec.)

- G+Smo (OMP_NUM_THREADS = 4)
- G+Smo
- MATLAB
- MATLAB (reduced)

Vase: 3.2, 11.0, 11.9, 9.0
Tooth: 2.6, 8.0, 8.9, 2.6
Duck: 14.9, 49.7, 57.9, 30.7
Component: 287.0, 367.2, 152.3, 80.8
Monkey: 646.4, 569.0, 222.9, 162.0
Multi-patch result: multipatch_tunnel.xml
Multi-patch result: yeti_footprint.xml
Compatible to multi-patch THB parameterization
Application: twin-screw rotary compressor
Application: twin-screw rotary compressor
Application: twin-screw rotary compressor
Thanks for your attention!

Q&A.

y.ji-1@tudelft.nl