Assignment Six: Koebe’s Iteration

David Gu

Yau Mathematics Science Center
Tsinghua University
Computer Science Department
Stony Brook University

gu@cs.stonybrook.edu

August 21, 2020
Figure: Circular slit map.
Gram–Schmidt Orthonormalization

Input: Poly annulus $M$, $\partial M = \gamma_0 - \gamma_1 - \cdots - \gamma_n$;
Output: $n - 1$ orthonormal non-exact harmonic 1-forms.

1. for $i = 0$ to $n$ do
2.   while true do
3.     generate a random non-exact harmonic 1-form $\omega_i$;
4.     for $j = 0$ to $i - 1$ do
5.       $w \leftarrow \langle \omega_i, \omega_j \rangle = \int_M \omega_i \wedge^* \omega_j$;
6.       $\omega_i \leftarrow \omega_i - w * \omega_j$;
7.     endfor
8.     $w \leftarrow \langle \omega_i, \omega_i \rangle$;
9.     if $w > 0.5$ then break;
10.   endwhile
11. $\omega_i \leftarrow \omega_i / \sqrt{w}$
12. endfor
Circular Slit Map Algorithm

Input: Poly annulus $M$, $\partial M = \gamma_0 - \gamma_1 - \cdots - \gamma_n$;
Output: Circular slit map $\varphi : M \to \mathbb{C}$.

1. Compute $n-1$ exact harmonic 1-forms, using assignment 3, $\omega_i$, such that $\omega_i = df_i$, $f_i$ is 1 on $\gamma_i$, and zero on $\gamma_j$, $j \neq i$, $f_i$ harmonic;
2. Gram–Schmidt orthonormalization;
3. Compute conjugate harmonic 1-forms, using Hodge star from assignment 3, obtain holomorphic 1-forms
   \[ \{ \omega_1 + \sqrt{-1}^* \omega_1, \omega_2 + \sqrt{-1}^* \omega_2, \ldots, \omega_{n-1} + \sqrt{-1}^* \omega_{n-1} \} \]
4. Solving linear equation $\omega := \sum_i \lambda_i (\omega_i + \sqrt{-1}^* \omega_i)$
   \[ \int_{\gamma_0} \omega = 1, \int_{\gamma_1} \omega = -1, \int_{\gamma_j} \omega = 0, j = 2, \ldots, n. \]
Choose a base point $p_0 \in M$, define the polar map:

$$\varphi(p) = \exp \left( 2\pi \int_{p_0}^{p} \omega \right).$$
Hole Filling Algorithm

Input: Poly annulus $M$, $\partial M = \gamma_0 - \gamma_1 - \cdots - \gamma_n$;
Output: A topological disk $\bar{M}$, such that all holes are filled.

1. $M_0 \leftarrow M$;
2. for $k = 0$ to $n$
   3. Compute a circular slit map, map the surface to the circular slit domain $f_k : M_k \rightarrow \mathbb{C}$, $\gamma_0$ and $\gamma_k$ are mapped to the exterior and interior circular boundary of $\mathbb{C}$;
   4. Generate a mesh $D_k$ using the inner boundary of $f_k(M_k)$ using Delaunay refinement mesh generation;
   5. Fill the inner circle of $f_k(M_k)$ to obtain $M_{k+1}$;
      \[ M_{k+1} \leftarrow f_k(M_k) \cup D_k. \]
3. endfor
6. $\bar{M} \leftarrow M_{n+1}$, return $\bar{M}$.  

David Gu (Stony Brook University)  
Computational Conformal Geometry  
August 21, 2020 6 / 26
Figure: Hole filling and slit map.
Koebe Iteration Method

Figure: Hole filling and slit map.
Figure: All holes are filled.
Koebe Iteration Algorithm

Input: A poly annulus $M$, $\partial M = \gamma_0 - \gamma_1 - \cdots - \gamma_n$
Output: A circular domain map $\varphi : M \rightarrow D$

1. Punch a hole at the $k$-th inner boundary;
2. Compute a circular slit map, to map the surface onto a canonical planar annulus;
3. Fill the inner circular hole;
4. Repeat step 4 through 6, each time punch a different hole, until the process converges.
Koebe Iteration Method
Koebe Iteration Method
Koebe Iteration Method
Koebe Iteration Method
Koebe Iteration Method
Koebe Iteration Method
Figure: Final result.
Instruction
Dependencies

1. ‘Delaunay’, Delaunay refinement for mesh generation.
2. ‘Hodge’, compute exact harmonic forms, random harmonic forms.
3. ‘MeshLib’, a mesh library based on halfedge data structure.
Directory Structure

- 3rdparty/Delaunay, headers, dlls, libs for Delaunay mesh generation;
- 3rdparty/Hodge, headers, dlls, libs for Hodge decomposition;
- koebeiteration/include, the header files for Koebe Iteration;
- koebeiteration/src, the source files for koebe iteration;
- data, Some data models and batch scripts;
- CMakeLists.txt, CMake configuration file;
- resources, snapshot for circular slit mapping results;
- textures, texture images needed.
Before you start, read README.md carefully, then go three the following procedures, step by step.

1. Install [CMake](https://cmake.org/download/).
2. Download the source code of the C++ framework.
3. Configure and generate the project for Visual Studio.
4. Open the .sln using Visual Studio, and compile the solution.
5. Finish your code in your IDE.
6. Run the executable program.
3. Configure and generate the project

1. open a command window
2. `cd Assignment_6_Koebe_Iteration_skeleton`
3. `mkdir build`
4. `cd build`
5. `cmake ..`
6. open `CCGHomework.sln` inside the build directory.
5. Finish your code in your IDE

- You need to modify the file: SlitMap.h, CircularSlitMap.cpp and HoleFiller.cpp
- search for comments “insert your code”
- Modify functions:
  1. \( C\text{SlitMap} \langle M \rangle :: \_slit\_map(\text{int}c1, \text{int}c2) \)
  2. \( \text{MeshLib} :: \text{polar\_map}(\text{CHodgeDecompositionMesh} * p\text{Mesh}) \)
  3. Gram-Schmidt orthonormalization

\[
\text{MeshLib} :: \text{calc\_holo\_1\_form\_open\_mesh}(\text{const}\text{std} :: \text{string}\& \text{input\_mesh}, \text{std} :: \text{vector}\langle \text{CHodgeDecompositionMesh}\star\rangle \& g\_\text{meshes}, \text{std} :: \text{vector}\langle \text{CHodgeDecompositionMesh}\star\rangle \& h\_\text{meshes}, \text{std} :: \text{string}\& \text{output\_mesh\_name})
\]
5. Finish your code in your IDE

- Modify functions:

```cpp
void MeshLib::punch_hole(CDTMesh& original_mesh,
                          CDTMesh& filled_mesh,
                          std::string& punched_mesh_name,
                          int id)
```
6. Run the executable program

Dynamic Linking Libraries

1. Construct a directory bin/;
2. Copy Delaunay.dll from 3rdparty/Delaunay/lib/windows/x64 to bin/;
3. Copy Hodge.dll from 3rdparty/Hodge/lib/windows/x64 to bin/;
4. Copy Koebeliteration.exe to bin/

Command

Go to data/boy_3_holes folder, click on koebe_iteration_test.bat.