C++とSFMLとQtで数値解析ソフトウェアを開発した話

23 Hogaraka

有限要素法って?

→構造を解析するために使われた数値解析手法

メリット　複雑な形状に対応できる

２次元ポアソン方程式をC++でQtとSFMLを用いて可視化したことを記す

数値解析について補足

数値解析とは手で解いて求める解析的な解を得るのが困難である場合に代わりに関数列(定数値の配列を関数とみたもの)を求める手法。

求める方程式の性質と導出

$$r(x) = \Delta u(x) + f(x)$$

$$\int_{\Omega} \omega(x) r(x) d\Omega = 0$$

有限要素法を重み付き残差法で解析
重み関数はガラーキン法を用いて
ポアソン方程式を求める。

有限要素法につかう最終的な式

$A_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j d\Omega$

$\mathbf{b}_i = \int_\Omega \phi_i f d\Omega + \int_{\Gamma_2} \phi_i p d\Gamma - \int_\Omega \nabla\phi_i \cdot \nabla u_g d\Omega$

$b_i = \frac{y_j - y_k}{2\Delta_e}$ $\quad A' = \begin{pmatrix} A^e_{kk} & A^e_{kl} & A^e_{km} \\ A^e_{jk} & A^e_{jl} & A^e_{jm} \\ A^e_{mk} & A^e_{ml} & A^e_{mm} \end{pmatrix}$

$c_i = \frac{x_k - x_j}{2\Delta_e}$ $\quad A^e_{ij} = \Delta_e \left(b^e_i b^e_j + c^e_i c^e_j \right)$

ここで i, j, kは A'における添え字で $j = (i+1)mod3, k = (i+2)mod3$

$A = \Sigma A'$

$$\mathbf{b}_e = \begin{pmatrix} \mathbf{b}_k \\ \mathbf{b}_l \\ \mathbf{b}_m \end{pmatrix} = \frac{\Delta_e}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} f_k \\ f_l \\ f_m \end{pmatrix}$$

尚、基本境界条件では$i = j$の場合上書きし、直接$A_{ij} = u_g$

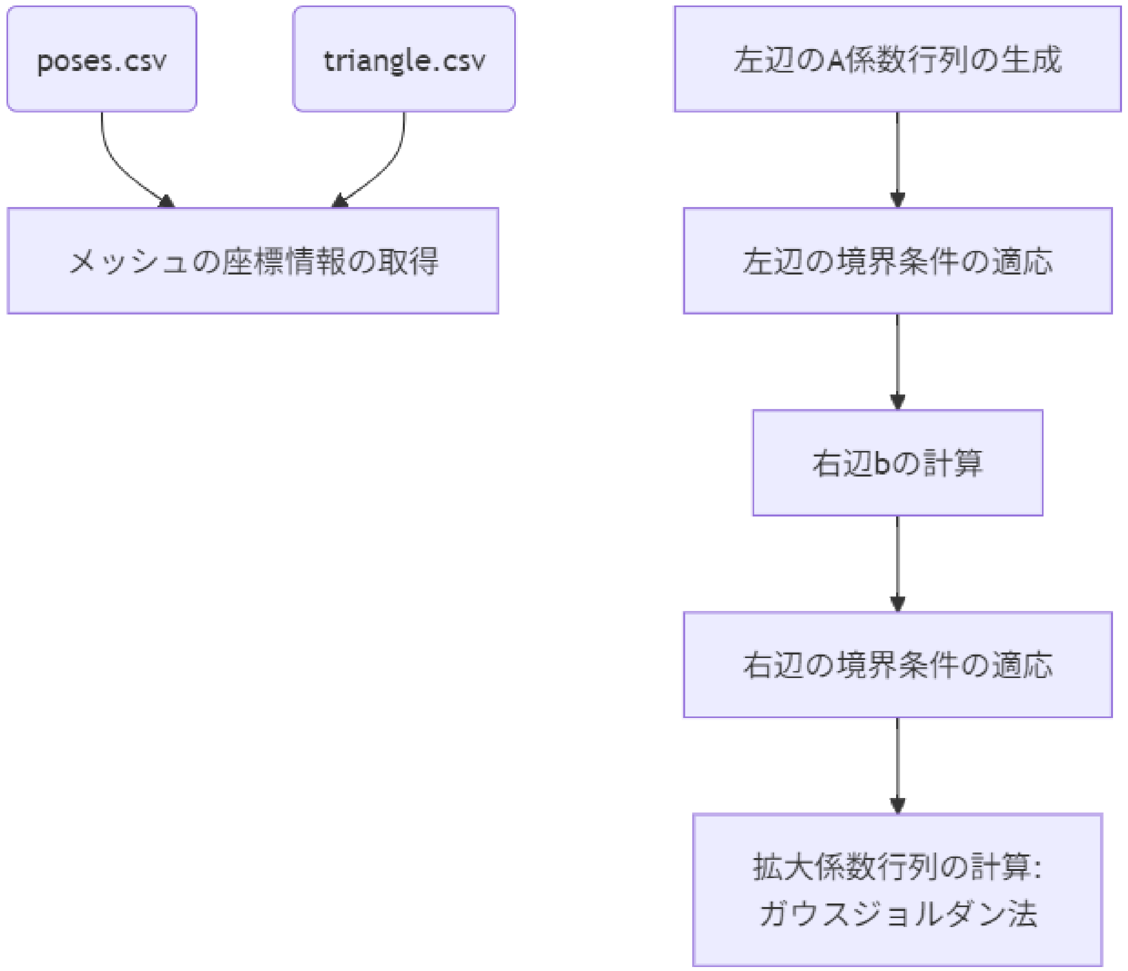
$i \neq j$で$A_{ij} = 0$とする

これをメッシュごとに計算して重ね合わせる → $AU = \mathbf{b}$

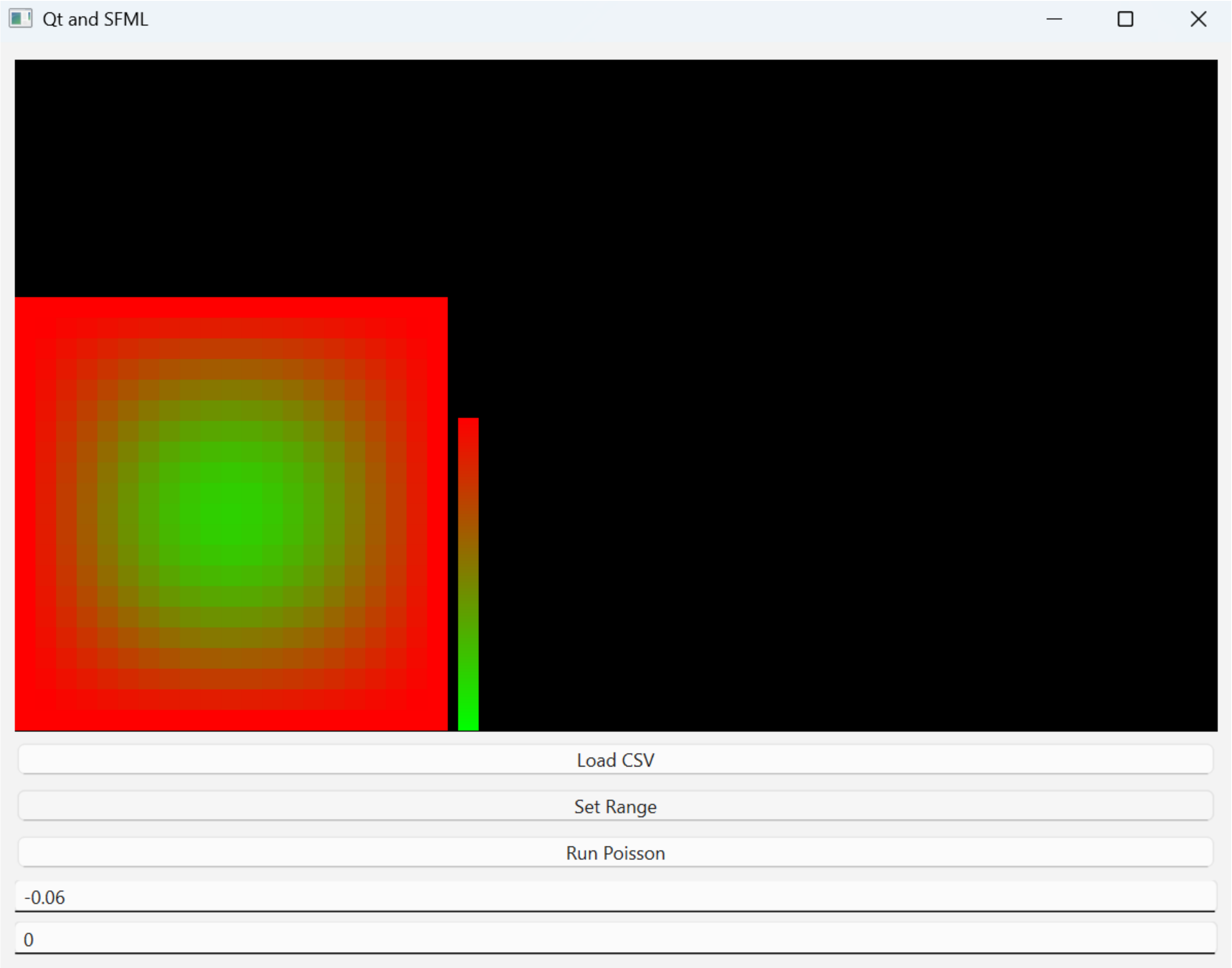
具体的な実装

メッシュ分割にはpythonを用いてnumpyで計算

残りはC++



実際の様子

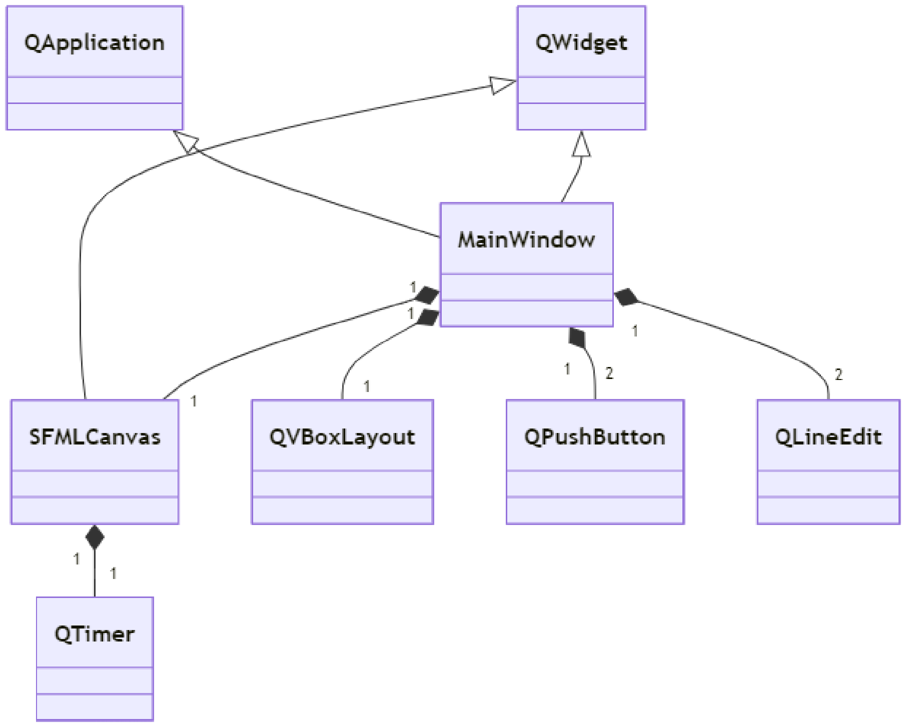


Qt,SFMLについてのアーキテクチャ

- QtはC++のクロスプラットホームGUI開発ライブラリ

→日本語表記やテキスト、ボタン処理などが実装しやすく

- SFMLはOpenGLのラッパーライブラリ。クロスプラットフォームでグラフィック処理が簡便に、かつOpenGLよりもレンダリングコンテキストや入出力の点でOS依存性が低い
- 右は簡単なクラス図

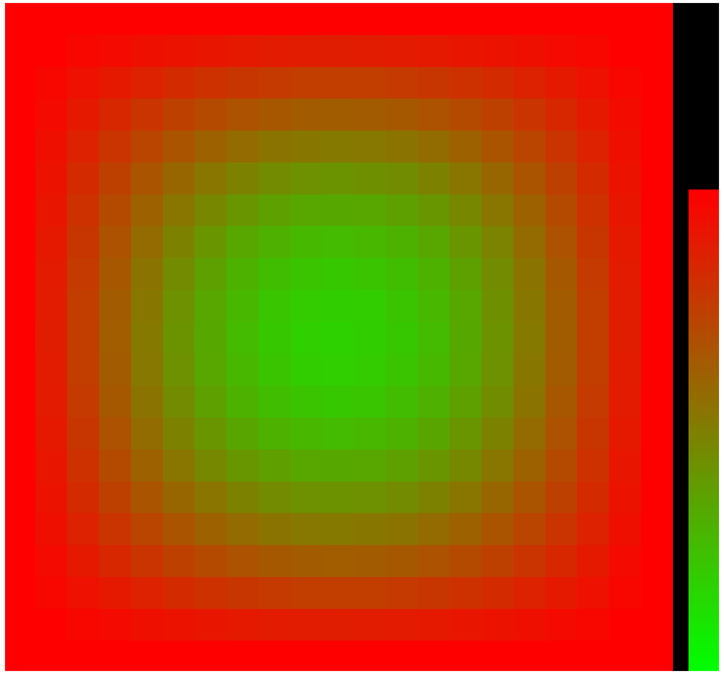


具体的な解析結果

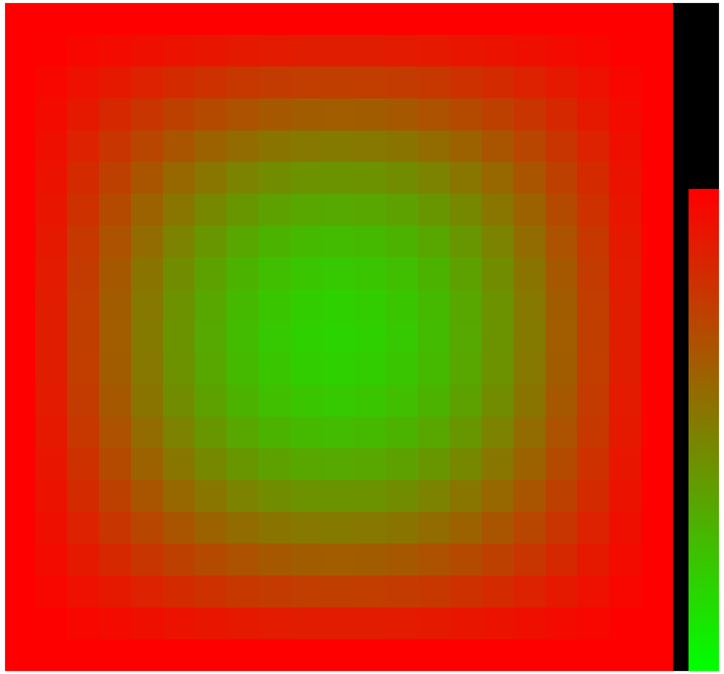
$$\Delta \mathbf{u} = sin(\pi x)sin(\pi y)$$

$[0,1]^2$ において 境界条件は $0(x = \pm 1, y = \pm 1)$

解析解は$u = -\frac{sin(\pi x)sin(\pi y)}{2\pi^2}$ である。



数値解析結果



真の値

両者ともに[-0.06,0]で色を表示

誤差はおよそ**0.7%**

↑小さい!!

Git hub

具体的なコードはこちらから

参考文献

有限要素法による流れのシミュレーション　計算力学学会
2015年