Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

universitätbonn

Exercises to Wissenschaftliches Rechnen I/Scientific Computing I (V3E1/F4E1)

Winter 2016/17

Prof. Dr. Martin Rumpf
Alexander Effland — Stefanie Heyden — Stefan Simon — Sascha Tölkes

Problem sheet 6

Please hand in the solutions on Tuesday December 6!

For the whole exercise sheet let

- $\Omega \subset \mathbb{R}^d$ be a polygonal domain with $d \leq 3$,
- $\mathcal{T}_h$ be a regular triangulation of Ω, i.e. $h(T) \leq c\rho(T)$,
- $V_h \subset H_0^{2,2}(\Omega)$ be a finite element space with $P_1 \subset \hat{P}$, s.t. all elements are affine equivalent to a reference element,
- $\mathcal{I}_h$ be the Lagrange interpolation operator,
- $f, f_h \in L^2(\Omega)$,
- $a(\phi, v) := \int_\Omega \Delta\phi\Delta v \,\mathrm{d}x$ for all $\phi, v \in H_0^{2,2}(\Omega)$, in particular $u|_{\partial\Omega} = 0$ and $\nabla u|_{\partial\Omega} = 0$ in the sense of traces,
- $u \in H_0^{2,2}(\Omega)$ be the weak solution of the biharmonic equation, i.e.
  $a(u, v) = \int_\Omega f \cdot v \,\mathrm{d}x$ for all $v \in H_0^{2,2}(\Omega)$,
- $u_h \in V_h$ be the discrete solution of the biharmonic equation, i.e.
  $a(u_h, v_h) = \int_\Omega f_h \cdot v_h \,\mathrm{d}x$ for $f \in V_h$ and all $v_h \in V_h$.

Note that for $d \leq 3$ the space $H^{2,2}(\Omega)$ compactly embeds into $C^0(\Omega)$, s.t. $\mathcal{I}_h$ is well defined.

Exercise 18 — 2 Points

Let $P_{k+1} \subset \hat{P}$ for $k \geq 0$. Use the Céa-Lemma to show that
$\|u - u_h\|_{2,2,\Omega} \leq Ch^k \|u\|_{k+2,2,\Omega}$.

Exercise 19 — 4 Points

Prove that for all $u \in H^{2,2}(\Omega)$ and for all $T \in \mathcal{T}_h$ and $E \in \mathcal{E}(T)$:

(i) $\|\phi - \mathcal{I}_h\phi\|_{0,2,E} \leq Ch(T)^{\frac{3}{2}} \|\phi\|_{2,2,T}$
(ii) $\|\phi - \mathcal{I}_h\phi\|_{1,2,E} \leq Ch(T)^{\frac{1}{2}} \|\phi\|_{2,2,T}$

Exercise 20 **4 Points**

(i) Show that for $\phi_h \in V_h$, $v \in H_0^{2,2}(\Omega)$ and $T \in \mathcal{T}_h$:

$$a(\phi_h, v) = \int_\Omega \Delta^2 \phi_h v_h \,\mathrm{d}x + \int_{\partial T} \Delta\phi_h \partial_n v_h - \nabla\Delta\phi_h \cdot n \; v_h \,\mathrm{d}a\,.$$

(ii) Prove the a posteriori error estimate

$$\|u - u_h\|_{2,2,\Omega} \le C\|f - f_h\|_{0,2,\Omega} + C\left(\sum_{T\in\mathcal{T}_h} \eta_T^2\right)^{\frac{1}{2}},$$

where

$$\eta_T^2 := \|h(T)^2(f_h - \Delta^2 u_h)\|_{0,2,T}^2 + \sum_{E\in\mathcal{E}_0(T)} \|h(T)^{\frac{1}{2}}\left[\Delta u_h\right]_E\|_{0,2,E}^2 + \|h(T)^{\frac{3}{2}}\left[\nabla\Delta u_h \cdot n_E\right]_E\|_{0,2,E}^2\,.$$

Hint: For (ii) follow the proof of theorem 2.3 and use the estimates from exercise 19.

Programming task 2

Consider a given triangular mesh $\mathcal{T}$ on a domain Ω. In this programming task, you will implement an adaptive grid refinement algorithm using the *longest edge bisection* method.
A common way to implement refinement algorithms on simplex grids is to use a concept called the *dart iterator*. This structure is defined via a triangle T, a node of that triangle (given by the local node index) and an edge of that triangle (given by the local edge index), as a "dart" pointing from the node along the edge .

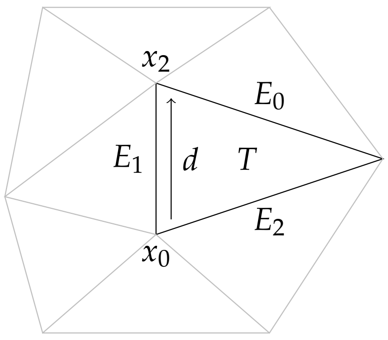
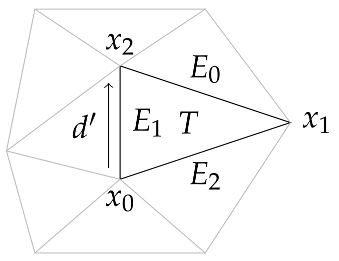


Figure 1: Example of a dart iterator d(T, 0, 1)

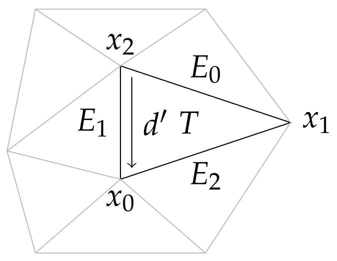
Figure 1 illustrates an example dart iterator d on T at node x_0 along edge E_1. Using this iterator three different types of actions are possible. These actions transform a dart iterator into another (well-defined) dart iterator:

flipTriangle() transforms $d(T, x, E)$ into the dart iterator $d'(T', x, E)$, where T' is the neighbor triangle of T for which $\bar{T} \cap \bar{T}' = E$ holds. Use **canFlipTriangle()** to check if this operation is possible, i. e. if T has a neighbor sharing E.

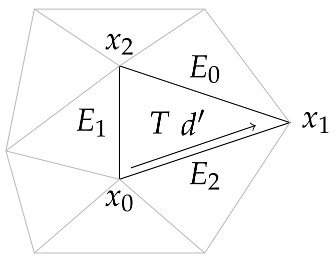
flipNode() results in the dart iterator $d'(T, x', E)$ along the same edge, but in the opposite direction.



(a) d.**flipTriangle()**



(b) d.**flipNode()**



(c) d.**d.flipEdge()**

Figure 2: $d' = d.\textbf{flip}\ \{\textbf{Triangle, Node, Edge}\}\textbf{()}$

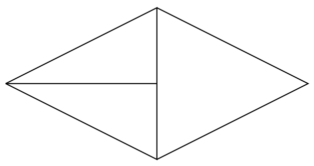
flipEdge() results in the dart iterator $d'(T, x, E')$ on the same triangle and point along the edge E' that shares the point x with E.

Figure 2 shows the effects of the three functions on the iterator d shown in Figure 1. The code fragment you have to complete is located in **lib/triangleMesh/adaptiveTriangMesh.h**.

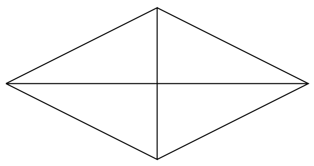
- Use the dart iterator to navigate the triangles that are marked for refinement and recursively modify the grid. Make sure only to subdivide the longest edge of each triangle (i. e. to subdivide longer edges, too, if necessary). Marked triangles are listed in **_markedForRefinement** and can be unmarked using the **unmark(**int element**)** function.

  Please note that the constructor of the class DartIterator in the C++ framework takes an additional constructor argument, the mesh. When working on an adaptive grid, *this can be passed as the first argument.

- The refinement rule used does not allow *hanging nodes*. Thus, neighbor elements may have to be refined even if they are not marked for refinement.



not allowed



allowed

- The GlobalIndex **addEdgeMidpoint(**const DartIterator &d**)** function can be used to subdivide an edge. The index returned is the index of the newly created vertex.

- Make sure to update and extend the **_neighbour_** vector. This vector contains vectors of integers storing the neighbors of a triangle on each edge.

- The index **IndexNotSet** can be used when an index should not refer to a triangle, index, or edge. This is e. g. useful, when a triangle does not have neighbors on all edges (triangles at the boundary of the domain).

Hint: It is helpful to split the refinement into several functions instead of using just the one function that already exists in the code.

On the lecture web site you can download an updated version of the C++ framework that already contains a program that can be used for testing.