MAT-67250 - Matematiikan Erikoistyö

Ville Räisänen
ville.raisanen@tut.fi
Tampereen Teknillinen Yliopisto

13. joulukuuta 2014

Sisältö

1 Elliptiset Tehtävät **2**
1.1 Dirichlet-tehtävä . . . 2
1.1.1 Heikko muotoilu . . . 2
1.1.2 Elementtimenetelmä . . . 2
1.1.3 A priori virherajat . . . 4
1.2 Robin-tehtävä . . . 7
1.2.1 Heikko muotoilu . . . 8
1.2.2 Yhteys vahvaan ratkaisuun . . . 8
1.2.3 Ratkaisun olemassaolo ja yksikäsitteisyys . . . 9
1.2.4 Virherajat . . . 9
1.3 Esimerkki Robin-tehtävästä . . . 10
1.3.1 Heikko muotoilu ja elementtimenetelmä . . . 11
1.3.2 Matriisien rakentaminen . . . 11
1.3.3 Testiesimerkki . . . 12
1.4 Adaptiivisuusstrategia . . . 13

2 Paraboliset Tehtävät **15**
2.1 Energia-estimaatit . . . 15
2.2 Elementtimenetelmä . . . 16
2.3 Esimerkki parabolisesta Robin-tehtävästä . . . 18
2.3.1 Elementtimenetelmä . . . 18
2.3.2 Aika-askellus . . . 19
2.3.3 Testiesimerkki . . . 19

A MATLAB-koodi **20**
A.1 Verkkotiedoston lukeminen . . . 21
A.2 Numeerinen integrointi . . . 25
A.3 Steady-state Robin-tehtävän toteutus . . . 29
A.4 Transientin Robin-tehtävän toteutus . . . 31

1 Elliptiset Tehtävät

1.1 Dirichlet-tehtävä

Tehtävän 10.6 ratkaisu edellyttää tehtävien 5.6-5.7 ratkaisemista, joten tarkastelemme aluksi reuna-arvotehtävää: Etsi $u \in C^2(\overline{\Omega})$ siten, että

$$\begin{aligned} -\nabla \cdot a\nabla u + \boldsymbol{b} \cdot \nabla u + cu &= f \quad \text{alueessa} \quad \Omega & (1)\\ u &= 0 \quad \text{reunalla} \quad \Gamma := \partial\Omega & (2) \end{aligned}$$

toteutuvat ja skalaarikentät a ja c sekä vektorikenttä b toteuttavat ehdot

$$\begin{aligned} a(x) &\geq a_0 > 0 & (3)\\ c(x) - \frac{1}{2}\nabla \cdot b(x) &\geq 0 & (4) \end{aligned}$$

kaikilla $x \in \Omega$.

1.1.1 Heikko muotoilu

Reuna-arvotehtävää $(1) - (4)$ vastaa heikko muotoilu: Etsi $u \in H_0^1(\Omega)$ siten, että

$$a(u,v) = L(v) \quad \forall v \in H_0^1(\Omega), \tag{5}$$

missä bilineaarimuoto $a : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ ja lineaarinen funktionaali $L : H_0^1(\Omega) \to \mathbb{R}$ määritellään

$$\begin{aligned} a(u,v) &:= \int_\Omega (a\nabla u \cdot \nabla v + v\boldsymbol{b} \cdot \nabla u + cuv), & (6)\\ L(v) &:= (f,v). & (7) \end{aligned}$$

Tällöin voidaan osoittaa (s. 32-34), että on olemassa vakiot $C_1, C_2 > 0$ siten, että kaikilla $u, v \in H_0^1(\Omega)$

$$\begin{aligned} a(u,u) &\geq a_0|u|_1^2 \geq C_1||u||_1^2, & (8)\\ a(u,v) &\leq C_2||u||_1||v||_1. & (9) \end{aligned}$$

Ratkaisun olemassaolo ja yksikäsitteisyys seuraavat *Lax-Milgramin lemmasta* (A.3, s. 230).

1.1.2 Elementtimenetelmä

Kappaleen 5.2 tavoin rajoitumme polygonaalisiin tehtäväalueisiin $\Omega \subset \mathbb{R}^2$, jotka jaetaan kolmiohin $\mathcal{T}_h = \{K\}$ siten, että

$$\overline{\Omega} = \bigcup_{K \in \mathcal{T}_h} K, \quad h = \max_{K \in \mathcal{T}_h} h_K : h_K = \operatorname{diam}(K) \tag{10}$$

toteutuvat, missä $\operatorname{diam}(K)$ on kolmion K suurin läpimitta. Kolmioiden kärkiä kutsumme *solmuiksi* ja oletamme, että kaksi kolmiota voivat leikata ainoastaan jakamalla kärkiä ja/tai kokonaisen reunan.

Kolmiointiin $\mathcal{T}_h$ liitämme paloittain affiinien funktioiden avaruuden

$$S_h \;:=\; \left\{v \in \mathcal{C}(\overline{\Omega}) : \forall K \in \mathcal{T}_h \; \exists a,b,c \in \mathbb{R} : v|_K(x,y) = a + bx + cy\right\} \subset H^1(\Omega) \qquad (11)$$

Tällöin jokaiselle kolmiolle $T \in \mathcal{S}_h$, funktion $v \in S_h$ rajoitus $v|_T$ kolmioon T määräytyy yksikäsitteisesti v:n arvoista kolmion kärkipisteissä. Edelleen koska funktion on oltava jatkuva eri kolmioiden välisten rajapintojen yli, funktio määräytyy yksikäsitteisesti $\overline{\Omega}$:ssa funktion arvoista solmupisteissä. Olkoon $\{P_i\}_{i=1}^{M_h} \subset \overline{\Omega}$ kolmioinnin $\mathcal{T}_h$ solmujen joukko. Täten valitsemme avaruuden S_h kantafunktioiksi funktiot $\{\Phi_i\}_{i=1}^{M_h} \subset S_h$ siten, että

$$\Phi_i(P_j) \;=\; \delta_{ij} \qquad (12)$$

toteutuu kaikilla $i,j = 1, \ldots, M_h$ missä δ on Kroneckerin delta.

Jaetaan solmut $\{P_i\}_{i=1}^{M_h}$ reunasolmuihin $\{P_i^I\}_{i=1}^{M_h^I}$ tehtäväalueen reunalla ja sisäsolmuihin $\{P_i^F\}_{i=1}^{M_h^F}$ tehtävaalueen sisällä.

Etsimme ratkaisua $u_h \in S_h$, joka kirjoitetaan muodossa

$$u_h \;=\; \sum_{i=1}^{M_h} U_i \Phi_i \qquad (13)$$

$$\;=\; \sum_{i=1}^{M_h^I} U_i^I \Phi_i^I + \sum_{i=1}^{M_h^F} U_i^F \Phi_i^F. \qquad (14)$$

Sillä elementtimenetelmä on Galerkin-menetelmä, käytämme testifunktioina ratkaisun esittämiseen käytettyjä funktioita. Tällöin päädymme heikosta muotoilusta (71) lineaariseen yhtälöryhmään

$$\sum_{j=1}^{M_h} U_j a(\Phi_j, \Phi_i) \;=\; L(\Phi_i), \quad i = 1, ..., M_h. \qquad (15)$$

Yhtälöryhmä (15) voidaan kirjoittaa matriisimuodossa

$$\big[A\big]\,\boldsymbol{U} \;=\; \boldsymbol{b}, \qquad (16)$$

missä $A \in \mathbb{R}^{M_h \times M_h}$ ja $\boldsymbol{b} \in \mathbb{R}^{M_h}$ ovat määritelty

$$A_{ij} \;:=\; a(\Phi_j, \Phi_i) \qquad (17)$$

$$b_i \;:=\; L(\Phi_i) \qquad (18)$$

Yhtälöryhmä (16) nyt vastaa reuna-arvotehtävää homogeenisella Neumann-reunaehdolla. Tehtävällä ei ole yksikäsitteistä ratkaisua sillä vakion lisääminen ratkaisuun tuottaa uuden ratkaisun (ks. Tehtävä 3.9). Mikäli potentiaali kiinnitetään jossain solmussa, ratkaisusta tulee yksikäsitteinen. Reunaehto (2) vastaa oletusta $\boldsymbol{U}^I = 0$.

1.1.3 A priori virherajat

Virherajoja todistaessa oletamme kirjan tapaan, että Ω on konveksin polygonin rajoittama alue. Olkoon $I_h : \mathcal{C}(\overline{\Omega}) \to S_h$ interpolaatiofunktio siten, että jokaiselle $v \in H^1(\Omega)$ jokaisessa solmupisteessä P_i

$$I_h v(P_i) \;:=\; v(P_i). \tag{19}$$

Tällöin saamme virherajat (ks. s. 61)

$$|I_h v - v|_1 \;\leq\; Ch||v||_2 \tag{20}$$
$$||I_h v - v|| \;\leq\; Ch^2||v||_2. \tag{21}$$

Problem 5.6. *(Galerkin's method) Let $a(\cdot,\cdot)$ and $L(\cdot)$ satisfy the assumptions of the Lax-Milgram lemma, i.e.,*

$$|a(v,w)| \;\leq\; C_1||v||_V||w||_V, \quad \forall v, w \in V, \tag{22}$$
$$a(v,v) \;\geq\; C_2||v||_V^2, \quad \forall v \in V, \tag{23}$$
$$|L(v)| \;\leq\; C_3||v||_V, \quad \forall v \in V. \tag{24}$$

Let $u \in V$ be the solution of

$$a(u,v) = L(v), \qquad \forall v \in V. \tag{25}$$

Let $\tilde{V} \subset V$ be a finite-dimensional subspace and let $\tilde{u} \in \tilde{V}$ be determined by Galerkin's method:

$$a(\tilde{u},v) \;=\; L(v), \quad \forall v \in \tilde{V}. \tag{26}$$

***(a)** Prove that (note that $a(\cdot,\cdot)$ may be non-symmetric)*

$$||\tilde{u} - u||_V \;\leq\; \frac{C_1}{C_2} \min_{\chi \in \tilde{V}} ||\chi - u||_V. \tag{27}$$

***(b, c)** Prove that if $a(\cdot,\cdot)$ is symmetric and $||v||_a = a(v,v)^{1/2}$, then*

$$||\tilde{u} - u||_a = \min_{\chi \in \tilde{V}} ||\chi - u||_a \quad \textit{and} \quad ||\tilde{u} - u||_V \leq \sqrt{\frac{C_1}{C_2}} \min_{\chi \in \tilde{V}} ||\chi - u||_V. \tag{28}$$

Todistus: Mielivaltaisella $\chi \in S_h$

$$a(u - \tilde{u}, \chi) = L(\chi) - L(\chi) = 0 \tag{29}$$

toteutuu, joten soveltamalla epäyhtälöä (22)

$$a(u - \tilde{u}, u - \tilde{u}) \;=\; a(u - \tilde{u}, u - \chi) \tag{30}$$
$$\;\leq\; C_1||u - \tilde{u}||_V||u - \chi||_V. \tag{31}$$

Toisaalta epäyhtälön (23) nojalla

$$a(u-\tilde{u},u-\tilde{u}) \;\geq\; C_2||u-\tilde{u}||_V^2. \tag{32}$$

Yhteensä siis

$$C_2||u-\tilde{u}||_V^2 \;\leq\; C_1||u-\tilde{u}||_V||u-\chi||_V, \tag{33}$$

josta väite **(a)** seuraa jakamalla. Olkoon $a(\cdot,\cdot)$ nyt symmetrinen. Soveltamalla yhtälöä (29), mielivaltaiselle $\chi \in \tilde{V}$

$$\begin{aligned} ||\chi-u||_a^2 &= ||\chi-\tilde{u}+\tilde{u}-u||_a^2 && (34)\\ &= ||\chi-\tilde{u}||_a^2+||\tilde{u}-u||_a^2+2a(\chi-\tilde{u},\tilde{u}-u) && (35)\\ &= ||\tilde{u}-\chi||_a^2+||\tilde{u}-u||_a^2 && (36)\\ &\geq ||\tilde{u}-u||_a^2, && (37) \end{aligned}$$

joka on väite **(b)**.
Soveltamalla epäyhtälöitä $(22)-(23)$, saamme kaikille $v \in V$

$$\sqrt{C_2}||v||_V \leq ||v||_a \leq \sqrt{C_1}||v||_V. \tag{38}$$

Sijoittamalla $v=\tilde{u}-u$ ja soveltamalla tulosta **(b)**

$$\sqrt{C_2}||\tilde{u}-u||_V \leq ||\tilde{u}-u||_a \leq ||\chi-u||_a \leq \sqrt{C_1}||\chi-u||_V, \tag{39}$$

josta väite **(c)** seuraa jakamalla. $\square$

Problem 5.7. *Consider the problem*

$$-\nabla\cdot(a\nabla u)+b\cdot\nabla u+cu=f \quad \textit{in} \quad \Omega, \quad \textit{with} \quad u=0 \quad \textit{on} \quad \Gamma. \tag{40}$$

from Sect. 3.5. Note that the presence of the convection term $b\cdot\nabla u$ makes the bilinear form non-symmetric.

(a) Formulate a finite element method for this problem and prove an error bound in the H^1-norm. Hint. See Problem 5.6.

(b) Prove an error bound in the L_2-norm. Hint: Modify the proof of Theorem 5.4 by using the auxiliary problem

$$\mathcal{A}^*\phi=-\nabla\cdot(a\nabla\phi)-b\cdot\nabla\phi+(c-\nabla\cdot b)\phi=c \quad \textit{in} \quad \Omega, \quad \phi=0 \quad \textit{on} \quad \Gamma, \tag{41}$$

instead of (5.46). *The operator $\mathcal{A}^*$ is the adjoint of $\mathcal{A}$, defined by $(\mathcal{A}v,w)=a(v,w)=(v,\mathcal{A}^*w)$ for all $v,w\in H^2\cap H_0^1$.*

(a) Elementtimenetelmän muotoilu on esitetty edellä. Ainoa ero kirjan kappaleessa 5 esitettyyn muotoiluun on monimutkaisempi bilineaarimuoto.

Lause 5.3 (yleistys) Olkoon u_h ja u ratkaisut tehtäviin (15) ja (5). Tällöin on olemassa $C > 0$ siten, että

$$|u_h - u|_1 \quad \leq \quad Ch||u||_2. \tag{42}$$

Todistus: Soveltamalla bilineaarimuodon coercive-ominaisuutta, tehtävän **5.6a** tulosta valinnalla $\chi = I_h u$, bilineaarimuodon rajoittuneisuutta, *Poincaren epäyhtälöä* (A.6) ja virherajaa (20), saamme

$$||u_h - u||_1 \leq C_1||u_h - u||_a \leq C_2||I_h u - u||_a \leq C_3||I_h u - u||_1 \leq C_4|I_h u - u|_1 \leq C_5 h||u||_2. \tag{43}$$

Väite seuraa huomioimalla $|v|_1 \leq ||v||_1$. $\square$

(b)
Lause 5.4 (yleistys) Olkoon Ω konveksi ja u_h ja u ratkaisut tehtäviin (15) ja (5). Tällöin on olemassa $C > 0$ siten, että

$$||u - u_h|| \quad \leq \quad Ch^2||u||_2. \tag{44}$$

Todistus: Määritellään bilineaarimuoto lisätehtävälle (41)

$$a^*(u,v) \quad := \quad \int_\Omega (a\nabla u \cdot \nabla v - v\boldsymbol{b} \cdot \nabla u + (c - \nabla \cdot b)uv) \tag{45}$$

$$= \quad \int_\Omega (a'\nabla u \cdot \nabla v + v\boldsymbol{b}' \cdot \nabla u + c'uv)\,, \tag{46}$$

missä $\boldsymbol{b}' = -\boldsymbol{b}$ ja $c' = c - \nabla \cdot \boldsymbol{b}$. Tällöin

$$c' - \frac{1}{2}\nabla \cdot \boldsymbol{b}' \quad = \quad c - \nabla \cdot \boldsymbol{b} + \frac{1}{2}\nabla \cdot \boldsymbol{b} = c - \frac{1}{2}\nabla \cdot \boldsymbol{b}, \tag{47}$$

joten ehdot $(3) - (4)$ toteutuvat ja lisätehtävällä on olemassa yksikäsitteinen ratkaisu. Olkoon ϕ ratkaisu tehtävään: Etsi $\phi \in H_0^1(\Omega)$ siten, että

$$a^*(\phi, w) = (e, w) \quad \forall w \in H_0^1(\Omega). \tag{48}$$

Nyt kaikille $\chi \in S_h$ toteutuu

$$a^*(\chi, u - u_h) \quad = \quad a(\chi, u) - a(\chi, u_h) = L(\chi) - L(\chi) = 0, \tag{49}$$

joten soveltamalla bilineaarimuodon a rajoittuneisuutta, Poincare'n lemmaa, (a)-kohtaa ja regulaarisuus-ominaisuutta (3.36), saamme

$$||e||^2 \quad = \quad a^*(\phi, e) \tag{50}$$

$$= \quad a^*(\phi - I_h\phi, e) \tag{51}$$

$$\leq \quad C_1||\phi - I_h\phi||_1||e||_1 \tag{52}$$

$$\leq \quad C_2|\phi - I_h\phi|_1|e|_1 \tag{53}$$

$$\leq \quad C_3 h||\phi||_2\,|e|_1 \tag{54}$$

$$\leq \quad C_4 h||e||\,|e|_1. \tag{55}$$

Jakamalla puolittain $||e||$:llä ja soveltamalla (a)-kohtaa, saamme

$$||u_h - u|| = ||e|| \leq C_1 h|e|_1 \leq C_2 h^2 ||u||_2. \tag{56}$$

□

Käyttämällä ominaisuutta (3.36), saadaan myös virheraja

$$||u - u_h|| \quad \leq \quad Ch^2||f|| \tag{57}$$

Määritellään *Ritz-projektio* $R_h : H_0^1(\Omega) \to S_h$ siten, että

$$a(R_h v - v, \chi) \quad = \quad 0, \quad \forall \chi \in S_h, \quad v \in H_0^1(\Omega). \tag{58}$$

Nyt mielivaltaiselle $v \in H_0^1(\Omega)$, $R_h v$ on ratkaisu tehtävään

$$a(R_h v, \chi) \quad = \quad L(\chi) := a(v, \chi), \tag{59}$$

joten Lauseen 5.4 yleistyksen nojalla

$$||R_h v - v|| \quad \leq \quad Ch^2||v||_2. \tag{60}$$

Käytämme tulosta Lauseen 10.1 yleistyksessä.

1.2 Robin-tehtävä

Problem 3.6) *Give a variational formulation of the problem*

$$-\Delta u = f, \quad \textit{with} \quad \frac{\partial u}{\partial n} + u = g \quad \textit{on} \quad \Gamma, \tag{61}$$

where $f \in L_2(\Omega)$ and $g \in L_2(\Gamma)$. Prove existence and uniqueness of a weak solution. Give an interpretation of the boundary condition with some problem in mechanics or physics.

Tarkastellaan reuna-arvotehtävää alueessa $\Omega \subset \mathbb{R}^d$: Etsi $u \in \mathcal{C}^2(\overline{\Omega})$ siten, että

$$-\nabla \cdot \nabla u \quad = \quad f, \quad \text{alueessa} \quad \Omega \tag{62}$$

$$u + \frac{\partial u}{\partial n} \quad = \quad g, \quad \text{reunalla} \quad \Gamma. \tag{63}$$

Kerrotaan Poisson-yhtälö (62) mielivaltaisella $v \in C^1(\overline{\Omega})$ ja integroidaan alueen Ω yli, jolloin päädymme yhtälöön

$$-\int_\Omega (\Delta u) v \quad = \quad \int_\Omega f v. \tag{64}$$

Soveltamalla Greenin toista identiteettiä, saamme

$$-\int_\Omega (\Delta u) v \quad = \quad \int_\Omega \nabla u \cdot \nabla v - \oint_\Gamma v \frac{\partial u}{\partial n} \tag{65}$$

Soveltamalla Robin-reunaehtoa (63) ja yhtälöä (64), päädymme yhtälöön

$$\int_\Omega \nabla u \cdot \nabla v + \oint_\Gamma uv = \int_\Omega fv + \oint_\Gamma gv. \tag{66}$$

1.2.1 Heikko muotoilu

Koska $C^1(\overline{\Omega})$ on tiheä $H^1(\Omega)$:ssa, (66) toteutuu myös testifunktioille $v \in H^1(\Omega)$: Jos olisi olemassa jokin $v \in H^1(\Omega)$ siten, että yhtälö (66) ei toteudu, on olemassa myös $\epsilon \in \mathbb{R} \setminus \{0\}$ siten, että

$$\int_\Omega \nabla u \cdot \nabla v + \oint_\Gamma uv = \int_\Omega fv + \oint_\Gamma gv + \epsilon. \tag{67}$$

Tiheyden nojalla voimme valita funktioita $w \in C^1(\overline{\Omega})$ siten, että erotuksen Sobolev-normi $||v - w||_1$ saadaan mielivaltaisen pieneksi. Valitsemalla funktion w Sobolev-normin suhteen tarpeeksi lähelle funktiota v päädymme ristiriitaiseen tilanteeseen, jossa (66) ei päde myöskään w:lle.

Siispä (66) pätee myös avaruuden $H^1(\Omega)$ testifunktioille. Toisaalta integraaleissa arvot tehtäväalueen reunalla Γ korvautuvat *trace-operaattorin* $\gamma : H^1(\Omega) \to L_2(\Gamma)$ antamilla arvoilla ja derivaatat Sobolev-avaruuksien rakentamiseen käytetyillä heikoilla derivaatoilla. Näin päädymme tehtävän (62) – (63) heikkoon muotoiluun: Etsi $u \in H^1(\Omega)$ siten, että

$$\int_\Omega \nabla u \cdot \nabla v + \oint_\Gamma \gamma u \gamma v \;\; = \;\; \int_\Omega fv + \oint_\Gamma g\gamma v \tag{68}$$

toteutuu kaikilla $v \in H^1(\Omega)$.

Määritellään bilineaarimuoto $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ ja lineaarinen funktionaali $L : H^1(\Omega) \to \mathbb{R}$ siten, että

$$a(u,v) \;\; := \;\; \int_\Omega \nabla u \cdot \nabla v + \oint_\Gamma \gamma u \gamma v, \tag{69}$$

$$L(v) \;\; := \;\; \int_\Omega fv + \oint_\Gamma g\gamma v \tag{70}$$

toteutuvat. Nyt heikko muotoilu voidaan kirjoittaa tiiviisti: Etsi $u \in H^1(\Omega)$ siten, että

$$a(u,v) = L(v) \qquad \forall v \in H^1(\Omega). \tag{71}$$

1.2.2 Yhteys vahvaan ratkaisuun

Oletetaan nyt käänteisesti, että $u \in C^2(\overline{\Omega})$ on ratkaisu tehtävään (71) ja sijoitetaan mielivaltainen $v \in H^1(\Omega)$ yhtälöön (64). Tällöin reunatermit katoavat ja saamme jälleen Greenin lausetta soveltamalla

$$\int_\Omega \nabla u \cdot \nabla v = -\int_\Omega (\Delta u)v + \oint_\Gamma \gamma \frac{\partial u}{\partial n} \gamma v = -\int_\Omega (\Delta u)v = \int_\Omega fv \tag{72}$$

tai

$$\int_\Omega (\Delta u + f)\, v \;\; = \;\; 0 \tag{73}$$

kaikilla $v \in H_0^1(\Omega)$. Edelleen jos $f \in \mathcal{C}(\Omega)$, $-\Delta u = f$ toteutuu ja kyseessä on *vahva ratkaisu.*

1.2.3 Ratkaisun olemassaolo ja yksikäsitteisyys

Ratkaisun olemassaolo ja yksikäsitteisyys seuraa *Lax-Milgramin Lemmasta* (A.3). Lemman soveltaminen edellyttää, että bilineaarimuoto (69) on coercive ja rajoitettu sekä, että lineaarinen funktionaali (70) on rajoitettu.

Cauchy-Schwarzin epäyhtälön ja *Trace-lauseen* (A.4) nojalla epäyhtälöt

$$\int_\Omega \nabla u \cdot \nabla v = (\nabla u, \nabla v)_{L_2} \leq ||\nabla u||\,||\nabla v|| \leq ||u||_1||v||_1, \tag{74}$$

$$\oint_\Gamma \gamma u \gamma v \leq ||\gamma u||_{L_2(\Gamma)}\,||\gamma v||_{L_2(\Gamma)} \leq C^2||u||_1||v||_1 \tag{75}$$

toteutuvat, joten voimme kirjoittaa epäyhtälön

$$a(u,v) \leq (1+C^2)||u||_1||v||_1. \tag{76}$$

Bilineaarimuoto a on siis rajoitettu. Sillä bilineaarimuoto on määritelty Sobolev-avaruudessa $H^1(\Omega)$ eikä $H^1_0(\Omega)$, emme voi käyttää *Poincaren epäyhtälöä* (A.6) osoittamaan, että a on coercive. Toisaalta soveltamalla Poincare-Friedrichs' epäyhtälöä (**Tehtävä 3.4**) ja avaruuden $C^1(\overline{\Omega})$ tiheyttä avaruudessa $H^1(\Omega)$

$$a(u,u) = ||\nabla u||^2 + ||u||^2_{L_2(\Gamma)} \geq \frac{1}{C^2}||u||^2_1, \tag{77}$$

joten a on coercive.

Soveltamalla Cauchy-Schwarzia ja Trace-lausetta, saamme epäyhtälön

$$\begin{aligned} L(v) &= (f,v)_{L_2(\Omega)} + (g,\gamma v)_{L_2(\Gamma)} && (78)\\ &\leq ||f||_{L_2(\Omega)}||v||_{L_2(\Omega)} + ||g||_{L_2(\Gamma)}||\gamma v||_{L_2(\Gamma)} && (79)\\ &\leq ||v||_1\left(||f||_{L_2(\Omega)} + C||g||_{L_2(\Gamma)}\right), && (80) \end{aligned}$$

joten lineaarinen funktionaali L on rajoitettu.

Lax-Milgramin lauseen nojalla ratkaisu on siis olemassa ja yksikäsitteinen. □

1.2.4 Virherajat

Tehtävä 5-11) *Formulate a finite element problem corresponding to the Robin problem in Problem 3.6. Prove error estimates.*

Robin-tehtävän FE-muotoilu saadaan yhtälöstä (71) rajoittumalla ratkaisun u_h ja testifunktioiden χ haussa avaruuteen S_h. Näin päädymme tehtävään: Etsi $u_h \in S_h$ siten, että

$$a(u_h,\chi) = L(\chi) \tag{81}$$

toteutuu kaikilla $\chi \in S_h$. Olkoon u ja u_h ratkaisut tehtäviin (71) ja (81), vastaavasti. Tällöin kaikilla $\chi \in S_h$

$$a(u_h - u, \chi) = a(u_h,\chi) - a(u,\chi) = L(\chi) - L(\chi) = 0. \tag{82}$$

Tällöin mielivaltaisella $\chi \in S_h$

$$\begin{aligned} ||\chi - u||_a^2 &= ||\chi - u_h + u_h - u||_a^2 && (83)\\ &= ||\chi - u_h||_a^2 + ||u_h - u||_a^2 + 2a(u_h - u, \chi - u_h) && (84)\\ &= ||\chi - u_h||_a^2 + ||u_h - u||_a^2 && (85)\\ &\geq ||u_h - u||_a^2, && (86) \end{aligned}$$

joten u_h minimoi a:n indusoiman energianormin avaruudessa S_h. Sijoittamalla $\chi = I_h u$ ja kirjoittamalla energianormit auki, saamme epäyhtälön

$$|u_h - u|_1^2 + ||\gamma(u_h - u)||_{L_2(\Gamma)}^2 \leq |I_h u - u|_1^2 + ||\gamma(I_h u - u)||_{L_2(\Gamma)}^2. \quad (87)$$

Trace-lauseen nojalla on olemassa $C_t > 0$ siten, että

$$\begin{aligned} ||\gamma(I_h u - u)||_{L_2(\Gamma)}^2 &\leq C_t^2 ||I_h - u||_1^2 && (88)\\ &= C_t^2 ||I_h u - u||^2 + C_t^2 |I_h u - u|_1^2. && (89) \end{aligned}$$

Sijoittamalla takaisin yhtälöön (87) ja soveltamalla ylärajoja (20)−(21) interpolaatiovirheelle, saamme epäyhtälön

$$\begin{aligned} |u_h - u|_1^2 + ||\gamma(u_h - u)||_{L_2(\Gamma)}^2 &\leq (1 + C_t^2)|I_h u - u|_1^2 + C_t^2 ||I_h u - u||^2. && (90)\\ &\leq (C_1 h^2 + C_2 h^4)||u||_2^2. && (91) \end{aligned}$$

Sillä molemmat termit vasemmalla puolella ovat ei-negatiivisia, virheylärajat pätevät molemmille termeille erikseen:

$$\begin{aligned} |u_h - u|_1^2 &\leq (C_1 h^2 + C_2 h^4)||u||_2^2, && (92)\\ ||\gamma(u_h - u)||_{L_2(\Gamma)}^2 &\leq (C_1 h^2 + C_2 h^4)||u||_2^2. && (93) \end{aligned}$$

Soveltamalla bilineaarimuodon a coercive-ominaisuutta yhtälöön (90), saamme

$$||u_h - u||_1^2 \leq (C_1 h^2 + C_2 h^4)||u||_2^2. \quad (94)$$

Sillä $h > 0$, on olemassa positiiviset C_3, C_4 siten, että

$$||u_h - u||_1 \leq (C_3 h + C_4 h^2)||u||_2. \quad (95)$$

Jakamalla normi osiin, voimme päätellä, että sama raja toimii myös L_2-normille ja Sobolev $|\cdot|_1$-seminormille. Verkon tihentäminen vastaa termin h pienentymistä, joten virherajaa dominoi asymptoottisesti pienemmän h:n potenssin termi.

1.3 Esimerkki Robin-tehtävästä

Tarkastellaan steady-state lämpötehtävää konvektio-reunaehdolla: Etsi $T \in C^2(\overline{\Omega})$

$$\begin{aligned} -\nabla \cdot k\nabla T &= p, && \text{alueessa} \quad \Omega, && (96)\\ -k\frac{\partial T}{\partial n} &= h(T - T_\infty), && \text{reunalla} \quad \Gamma. && (97) \end{aligned}$$

Skalaarikenttä $k \geq k_0 > 0$ tällöin kuvaa *lämmönjohtavuutta* ja skalaarit $h > 0$ ja $T_\infty > 0$ *konvektiokerrointa rajapinnalla* Γ ja *ympäristön lämpötilaa*, vastaavasti.

1.3.1 Heikko muotoilu ja elementtimenetelmä

Lähdetään liikkeelle heikosta muotoilusta: Etsi $T \in H^1(\Omega)$ siten, että

$$a(T, v) \;=\; L(v) \tag{98}$$

toteutuu kaikilla $v \in H^1(\Omega)$, missä bilineaarimuoto $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ määritellään

$$a(T, v) \;:=\; \int_\Omega k\nabla T \cdot \nabla v + \oint_\Gamma h\gamma T\, \gamma v \tag{99}$$

ja lineaarinen funktionaali $L : H^1(\Omega) \to \mathbb{R}$ määritellään

$$L(v) \;:=\; \int_\Omega fv + \oint_\Gamma hT_\infty \gamma v. \tag{100}$$

Haemme elementtimenetelmällä ratkaisua $T_h \in S_h$

$$T_h \;=\; \sum_{i=1}^{M_h^I} U_i^I \Phi_i^I + \sum_{i=1}^{M_h^F} U_i^F \Phi_i^F \tag{101}$$

siten, että

$$a(T_h, v) \;=\; f(v) \tag{102}$$

toteutuu kaikilla $v \in S_h$. Määritellään *jäykkyys-* ja *(reuna)massa-matriisit* $\mathcal{S}, \mathcal{N} \in \mathbb{R}^{M_h \times M_h}$:

$$\mathcal{S}_{ij} \;:=\; \int_\Omega k\nabla\Phi_j \cdot \nabla\Phi_i, \tag{103}$$

$$\mathcal{N}_{ij} \;:=\; \int_\Gamma h\gamma\Phi_j\gamma\Phi_i. \tag{104}$$

ja vektorit $\boldsymbol{b}, \boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}^{M_h}$:

$$b_i \;:=\; \int_\Omega p\Phi_i + \oint_\Gamma hT_\infty\gamma\Phi_i =: f_i + g_i. \tag{105}$$

Jaetaan matriisit ja vektorit lohkoihin vastaten sisä- ja reunasolmuja, jolloin päädymme yhtälöryhmään

$$\begin{bmatrix} \mathcal{S}_{FF} & \mathcal{S}_{FI} \\ \mathcal{S}_{IF} & \mathcal{S}_{II} + \mathcal{N}_{II} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^F \\ \boldsymbol{U}^I \end{bmatrix} \;=\; \begin{bmatrix} \boldsymbol{f}_F \\ \boldsymbol{g}_I \end{bmatrix}. \tag{106}$$

1.3.2 Matriisien rakentaminen

MATLAB-implementaatiossa (ks. `stiff_global_gauss.m`) kunkin globaalin matriisin rakentaminen aloitetaan luomalla verkon jokaiselle kolmiolle kolmion kantafunktioihin rajoittunut elementtimatriisi. Elementtimatriisien alkiot sekä alkioita vastaavat sijainnit globaalissa matriisissa kootaan listaan, josta MATLAB osaa nopeasti rakentaa harvan esityksen matriisille. Integraalit kunkin elementin yli lasketaan numeerisesti integroimalla *referenssielementille*

koordinaattimuunnoksella palautettu integrandi. Gauss-pisteiden sijainnit referenssielementillä sekä pisteiden painot ovat annettu skriptissä `gaussdata.m`.

Olkoon $K_0 := \{(x,y) \in [0,1]^2 \subset \mathbb{R}^2 : y \leq 1-x\}$ referenssielementti ja J_K Jacobin matriisi affiinille sijoituskuvaukselle referenssielementiltä K_0 elementille K. Olkoon $W_g, g = 1, ..., N_g$ Gauss-pisteiden painot ja merkitään alaindeksillä g funktioiden arvoja kussakin Gauss-pisteessä g. Tällöin, mielivaltaisen elementin $K \in \mathcal{T}_h$ osuus jäykkyysmatriisin (103) mielivaltaisesta alkiosta voidaan kirjoittaa ja approksimoida numeerisesti integroimalla

$$\begin{aligned} \mathcal{S}_{ij}^K &= \int_K k \nabla \Phi_j \cdot \nabla \Phi_i && (107) \\ &= |J_K| \int_{K_0} k \nabla \Phi_j^T J_K^{-1} J_K^{-T} \nabla \Phi_i && (108) \\ &\approx |J_t| \sum_{g=1}^{N_g} W_g k_g \nabla \Phi_{jg}^T J_K^{-1} J_k^{-T} \nabla \Phi_{ig} && (109) \end{aligned}$$

Mikäli f ja p voidaan olettaa vakioiksi kussakin elementissä sekä h vakioksi kunkin reunan Γ kaarella, matriisi $\mathcal{M}$ sekä vektorit $\boldsymbol{b}, \boldsymbol{f}$ ja $\boldsymbol{g}$ voidaan laskea helposti käsin integroimalla.

Oletetaan, että p on vakio kussakin kolmiossa $K \in \mathcal{T}_h$. Olkoon $\mathcal{T}_h^i \subset \mathcal{T}_h$ solmun i sisältävien kolmioiden joukko. Tällöin

$$f_i = \sum_{K \in \mathcal{T}_h^i} p_K \int_K \Phi_i = \sum_{K \in \mathcal{T}_h^i} \frac{p_K}{3} A_K, \quad (110)$$

missä A_K on kolmion K pinta-ala. Vastaavasti jos $\mathcal{E}_h^i$ on solmun i sisältävien reunan Γ kaarien joukko,

$$\begin{aligned} \mathcal{N}_{ij} &= \sum_{e \in \mathcal{E}_h^i} h \int_e \gamma \Phi_i \gamma \Phi_j && (111) \\ g_i &= h T_\infty \sum_{e \in \mathcal{E}_h^i} \int_e \gamma \Phi_i = h T_\infty \sum_{e \in \mathcal{E}_h^i} \frac{l_e}{2}, && (112) \end{aligned}$$

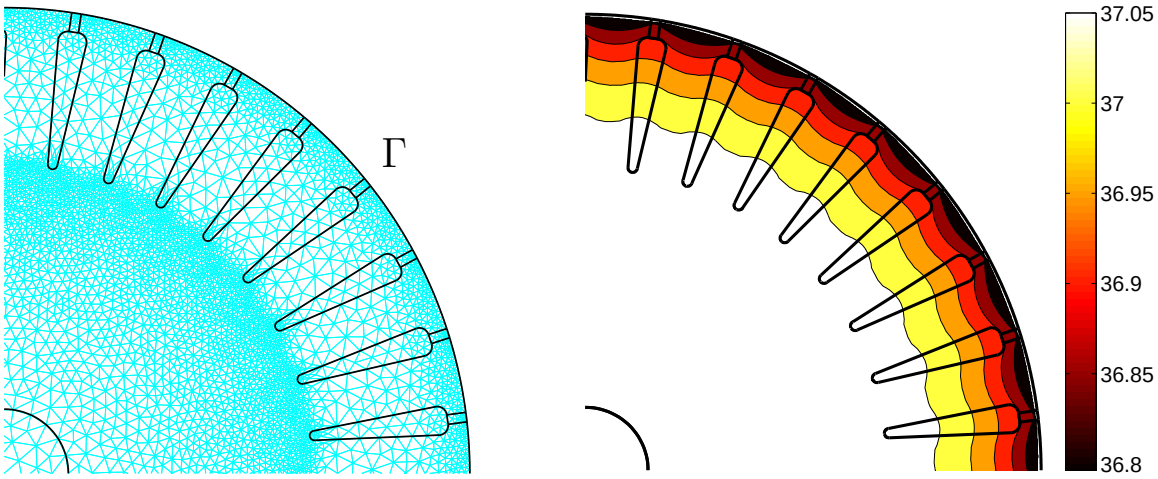
missä l_e on reunan e pituus.

1.3.3 Testiesimerkki

Sovellamme muodostettua steady-state lämpötehtävää (96) – (97) oikosulkumoottorin roottorin lämpötilan ennustamiseen (ks. Kuva 1). Roottorin geometria perustuu julkaisussa[1] käytettyyn testiesimerkkiin. Testiesimerkin MATLAB-toteutus on esitetty skriptissä `robin_steadystate.m` (ks. Liite A).

Konvektio-reunaehdolla (97) mallinnetaan roottorin ulkoreunalla Γ tapahtuvaa lämmönsiirtoa "ympäristöön", johon liitetään lämpötila $T_\infty = 20\,^\circ C$. Kentällä $p \approx 431.31\,\mathrm{kW/(m^2 \cdot m)}$

[1] A. Arkkio. Analysis of Induction Motors Based on the Numerical Solution of the Magnetic Field and Circuit Equations. ACTA POLYTECHNICA SCANDINAVICA, PhD thesis, 1987.



Kuva 1: Verkko, geometria ja ratkaisu tehtäväalueen neljänneksessä.

kuvataan pyörrevirtahäviötiheyttä pituusyksikköä kohti roottorin palkeissa, joka vastaa kokonaishäviötä yksikköpituutta kohti $P = 1000\,\mathrm{W/m}$. Konvektiokertoimeksi on valittu keinotekoisesti realistista kokoluokkaa oleva arvo $h = 130\,\mathrm{W/Km^2}$, joka esiintyy julkaisussa[2]. Realistisemman mallin tulisi huomioida ilmavälin lämpötilan ja konvektiokertoimen riippuvuus roottorin pyörimisnopeudesta sekä tarkastelupisteen sijainnista roottorin pinnalla.

Vertailutulos saadaan huomioimalla, että steady-state tilassa pinnalta Γ poistuva teho on yhtä suuri koko tehtäväalueessa tapahtuvien häviöiden kanssa. Ts.

$$\int_\Gamma h(T - T_\infty) \quad = \quad \int_\Omega p := P. \tag{113}$$

Olkoon r ympyrän Γ säde. Approksimoimalla pinnan Γ lämpötila paikan suhteen vakioksi, (113) yksinkertaistuu muotoon

$$2\pi r h(T - T_\infty) \quad = \quad P, \tag{114}$$

josta voimme ratkaista pinnan lämpötilaksi

$$T \quad = \quad T_\infty + \frac{P}{2\pi r h} = 20^\circ\mathrm{C} + \frac{1000\,\mathrm{W/m}}{2\pi \cdot 0.0725\,\mathrm{m} \cdot 130\,\mathrm{W/Km^2}} \approx 36.77^\circ\mathrm{C}. \tag{115}$$

Elementtimenetelmällä ratkaistujen reunan Γ solmupisteiden arvot vaihtelevat välillä $36.77^\circ\mathrm{C} - 36.87^\circ\mathrm{C}$. Tulos siis täsmää erittäin hyvin. Edelleen molemmilla ratkaisutavoilla saaduille ratkaisuille erotukset $T - T_\infty$ ovat lineaarisia häviötiheys-kentän p suhteen, joten ratkaisut täsmäävät riippumatta häviötiheyden skaalauksesta.

1.4 Adaptiivisuusstrategia

Yksinkertainen adaptiivisuusstrategia seuraa lauseesta 5.6:

[2] D. Howey, P. Childs, A. Holmes, Air-Gap Convection in Rotating Electrical Machines, *IEEE Transactions on Industrial Electronics*, vol. 59, no. 3, 2010.

Theorem 5.6. Assume that Ω is a convex polygonal domain in the plane. Let u_h and u be the solutions of (5.26) and (5.24), respectively. Then

$$||u_h - u|| \leq C \left(\sum_{K \in \mathcal{T}_h} R_K^2 \right)^{1/2}, \tag{116}$$

where

$$R_K = h_K^2 ||\mathcal{A}u_h - f||_K + h_K^{3/2} ||a[n \cdot \nabla u_h]||_{\partial K \setminus \Gamma}, \tag{117}$$

and $[n \cdot \nabla u_h]$ denotes the jump across ∂K in the normal derivative $n \cdot \nabla u_h$.

Toisaalta lauseen yläraja (116) sisältää ennalta tuntemattoman kertoimen C, joten emme saa virheelle eksplisiittistä ylärajaa emmekä voi asettaa toleranssia kokonaisvirheelle $||u_h - u||$. Toisaalta laskemalla jokaiselle kolmiolle kertoimet R_K yhtälöstä (117), voimme päätellä ne kolmiot, jotka kasvattavat ylärajaa (116) eniten. Yksinkertainen adaptiivisuusstrategia on siis toistaa ratkaisua ja jakaa ne kolmiot K osiin, joita vastaa suurin virhe R_K.

2 Paraboliset Tehtävät

Tarkastellaan parabolista reuna-arvotehtävää: Etsi $u \in C^2(\overline{\Omega})$ siten, että

$$\frac{\partial u}{\partial t} - \nabla \cdot (a\nabla u) + b \cdot \nabla u + cu = f, \qquad \text{alueessa} \quad \Omega \tag{118}$$

$$u = 0, \qquad \text{reunalla} \quad \Gamma, \tag{119}$$

$$u = v, \qquad \text{alkuhetkellä} \quad t = 0 \tag{120}$$

toteutuvat, missä $a \geq a_0 > 0$ ja $c - \frac{1}{2}\nabla \cdot b \geq 0$. Vastaava heikko muotoilu voidaan kirjoittaa: Etsi $u \in H_0^1(\Omega)$ siten, että

$$(u_t, \varphi) + a(u, \varphi) = (f, \varphi), \quad \forall \varphi \in H_0^1(\Omega) \tag{121}$$

toteutuu, missä bilineaarimuoto $a : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ määritellään

$$a(u, v) \;:=\; \int_\Omega (a\nabla u \cdot \nabla v + v\boldsymbol{b} \cdot \nabla u + cuv) \tag{122}$$

2.1 Energia-estimaatit

Problem 8.8. *Show estimates analogous to those of Theorem 8.5 when the elliptic term* $-\Delta u$ *in* (8.21) *has been replaced by* $\mathcal{A} = -\nabla \cdot (a\nabla u) + b \cdot \nabla u + cu$ *as in Sect. 3.5.*

Theorem 8.5. *Let* $u(t)$ *satisfy (8.29) and (8.30), vanish on* Γ*, and be appropriately smooth for* $t \geq 0$*. Then there is a constant* C *such that, for* $t \geq 0$*,*

$$||u(t)||^2 + \int_0^t |u(s)|_1^2 \, ds \;\leq\; ||v||^2 + C \int_0^t ||f(s)||^2 \, ds, \tag{123}$$

$$|u(t)|_1^2 + \int_0^t ||u_t(s)||^2 \, ds \;\leq\; |v|_1^2 + \int_0^t ||f(s)||^2 \, ds. \tag{124}$$

Lause 8.5 (yleistys). Olkoon $u \in H_0^1(\Omega)$ yhtälöt (121) ja (120) toteuttava funktio. Tällöin on olemassa vakio $C > 0$ siten, että

$$||u(t)||^2 + (2a_0 - 1) \int_0^t |u(s)|_1^2 \, ds \;\leq\; ||v||^2 + C \int_0^t ||f(s)||^2 \, ds, \tag{125}$$

$$a_0 |u(t)|_1^2 + \int_0^t ||u_t(s)||^2 \, ds \;\leq\; a(v, v) + \int_0^t \left(||f||^2(s) + (u, \boldsymbol{b} \cdot \nabla u_t(s)) \right) \, ds \tag{126}$$

toteutuvat kaikille $t > 0$.

Todistus: Sijoitetaan $\varphi = u$ yhtälöön (121), jolloin saamme

$$(u_t, u) \;=\; \int_\Omega u_t u = \frac{1}{2} \frac{d}{dt} ||u||^2. \tag{127}$$

Soveltamalla Poincare'n epäyhtälöä sekä epäyhtälöä $2ab \leq a^2 + b^2$, saamme

$$|(f, u)| \;\leq\; ||f|| \, ||u|| \leq \frac{1}{2} |u|_1^2 + \frac{1}{2} C^2 ||f||^2. \tag{128}$$

Bilineaarioperaattorille a toteutuu (ks. s. 34)

$$a(u,u) \;\geq\; a_0|u|_1^2. \tag{129}$$

Yhteensä siis

$$\frac{1}{2}\frac{d}{dt}||u||^2 + \left(a_0 - \frac{1}{2}\right)|u|_1^2 \leq \frac{1}{2}C^2||f||^2. \tag{130}$$

Kertomalla kahdella ja integroimalla ajan suhteen välin $[0,t]$ yli ja uudelleenjärjestelemällä saamme epäyhtälön (125). Valitsemalla $a = a_0 = 1$ päästään alkuperäiseen väitteeseen, joten kyseessä on yleistys.

Sijoitetaan $\varphi = u_t$ bilineaarimuotoon (122), jolloin saamme

$$\begin{aligned} a(u,u_t) &= \int_\Omega (a\nabla u\cdot\nabla u_t + u_t\boldsymbol{b}\cdot\nabla u + cuu_t) && (131)\\ &= \frac{1}{2}\frac{d}{dt}a(u,u) - \frac{1}{2}(u,\boldsymbol{b}\cdot\nabla u_t). && (132)\end{aligned}$$

Sijoittamalla $\varphi = u_t$ yhtälön (121) muihin termeihin, saamme

$$\begin{aligned} (u_t,u_t) &= ||u_t||^2, && (133)\\ |(f,u_t)| &\leq ||f||\,||u_t|| \leq \frac{1}{2}||f||^2 + \frac{1}{2}||u_t||^2. && (134)\end{aligned}$$

Yhteensä

$$||u_t||^2 + \frac{1}{2}\frac{d}{dt}a(u,u) - \frac{1}{2}(u,\boldsymbol{b}\cdot\nabla u_t) \;\leq\; \frac{1}{2}||u_t||^2 + \frac{1}{2}||f||^2. \tag{135}$$

Kertomalla kahdella, integroimalla ajan suhteen välin $[0,t]$ yli ja uudelleenjärjestelemällä, saamme epäyhtälön

$$a(u(t),u(t)) + \int_0^t ||u_t(s)||^2\,ds \;\leq\; a(v,v) + \int_0^t \left(||f||^2(s) + (u,\boldsymbol{b}\cdot\nabla u_t(s))\right)\,ds. \tag{136}$$

Soveltamalla epäyhtälöä (129), saamme yhtälön (126). Valitsemalla $a = a_0 = 1, b = c = 0$ päästään alkuperäiseen väitteeseen, joten kyseessä on yleistys. □

2.2 Elementtimenetelmä

Parabolista tehtävää (118) – (120) vastaava FE-tehtävä saadaan jälleen heikosta muodosta (121) rajoittamalla ratkaisun ja testifunktioiden haku avaruuteen S_h. Näin päädytään tehtävään: Etsi $u_h \in S_h$ siten, että

$$(u_{ht},\chi) + a(u_h,\chi) = (f,\chi), \quad \forall\chi S_h. \tag{137}$$

Problem 10.6. *Show error estimates analogous to those of Theorem 10.1 when the elliptic term $-\Delta u$ in (10.1) has been replaced by $\mathcal{A} = -\nabla\cdot(a\nabla u) + b\cdot\nabla u + cu$ as in Sect. 3.5. Hint: See Problems 5.7 and 8.8.*

Lause 10.1 (yleistys) Olkoon u_h ja u ratkaisut tehtäviin (137) ja (121). Tällöin

$$||u_h(t) - u(t)|| \leq ||v_h - v|| + Ch^2 \left(||v||_2 + \int_0^t ||u_t||_2 \, ds \right) \tag{138}$$

Todistus: Kirjoitetaan

$$u_h - u \;=\; (u_h - R_h u) + (R_h u - u) =: \theta + \rho. \tag{139}$$

Nyt yhtälön (60) ja kolmioepäyhtälön nojalla

$$||\rho|| \;\leq\; Ch^2 ||u||_2 = Ch^2 \left|\left| v + \int_0^t u_s(s) \, ds \right|\right|_2 \leq Ch^2 \left(||v||_2 + \int_0^t ||u_s(s)||_2 \, ds \right). \tag{140}$$

Toisaalta yhtälöiden (137), (58) ja (139) nojalla

$$\begin{aligned}
(\theta_t, \chi) + a(\theta, \chi) &= (u_{ht}, \chi) + a(u_h, \chi) - (R_h u_t, \chi) - a(R_h u, \chi) && (141)\\
&= (f, \chi) - (R_h u_t, \chi) - a(u, \chi) && (142)\\
&= (u_t - R_h u_t, \chi) && (143)\\
&= -(\rho_t, \chi). && (144)
\end{aligned}$$

Ts. θ toteuttaa FE-yhtälöt (137) lähdetermille $-\rho_t$. Ts.

$$(\theta_t, \chi) + a(\theta, \chi) = -(\rho_t, \chi), \quad \forall \chi \in S_h. \tag{145}$$

Sijoittamalla $\chi = \theta$,

$$||\theta|| \frac{d}{dt} ||\theta|| + a(\theta, \theta) \;\leq\; ||\rho_t|| ||\theta||. \tag{146}$$

Nyt a:n coercive-ominaisuuden nojalla $a(\theta, \theta) > 0$, joten jakamalla $||\theta||$:lla, saamme

$$\frac{d}{dt} ||\theta|| \;\leq\; ||\rho_t||. \tag{147}$$

Integroimalla puolittain välin $[0, t]$ yli,

$$\begin{aligned}
||\theta(t)|| - ||\theta(0)|| &= ||u_h(t) - R_h u(t)|| - ||v_h - R_h v|| && (148)\\
&\leq ||\rho(t)|| - ||\rho(0)| && (149)\\
&= ||R_h u(t) - u(t)|| - ||R_h v - v|| && (150)\\
&\leq ||R_h u(t) - u(t)|| && (151)\\
&= ||\rho(t)||. && (152)
\end{aligned}$$

Soveltamalla yhtälöitä (60) ja (21), saamme

$$||\theta(0)|| \;=\; ||v_h - R_h v|| \leq ||v_h - v|| + ||R_h v - v|| \leq Ch^2 ||v||_2, \tag{153}$$

joten myös

$$||\theta(t)|| \;\leq\; Ch^2 ||v||_2 + ||\rho(t)||. \tag{154}$$

Soveltamalla ρ:lle saatua estimaattia, saamme väitteen. $\square$

2.3 Esimerkki parabolisesta Robin-tehtävästä

Tarkastelemme parabolista reuna-arvotehtävää: Etsi $T \in C^2(\overline{\Omega})$ siten, että

$$\begin{aligned} c\frac{\partial T}{\partial t} - \nabla \cdot k\nabla T &= p, \qquad \text{alueessa} \quad \Omega, && (155)\\ -k\frac{\partial T}{\partial n} &= h(T - T_\infty), \quad \text{reunalla} \quad \Gamma, && (156)\\ T(0) &= T_\infty, \qquad \text{alueessa} \quad \Omega, && (157) \end{aligned}$$

missä skalaarikenttä $c \geq c_0 > 0$ kuvaa materiaalin lämpökapasiteettia tilavuuden suhteen.

Tehtävää $(155) - (156)$ vastaava heikko muotoilu voidaan kirjoittaa: Etsi $T \in H^1(\Omega)$ siten, että

$$(c\partial^{\,t} T(t), v) + a(T(t), v) = L(v), \tag{158}$$

toteutuu kaikilla $v \in H^1(\Omega)$ missä bilineaarimuoto $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ ja lineaarinen funktionaali $L : H^1(\Omega) \to \mathbb{R}$ määritellään

$$\begin{aligned} a(T, v) &:= \int_\Omega k\nabla T \cdot \nabla v + \oint_\Gamma h\gamma T\, \gamma v, && (159)\\ L(v) &:= (f, v) + \oint_\Gamma hT_\infty \gamma v. && (160) \end{aligned}$$

Kappaleiden 2.1-2.2 tehtävään päästään jakamalla integrandit skalaarikentällä c.

2.3.1 Elementtimenetelmä

Sijoittamalla lausekkeen (158) ensimmäiseen termiin ratkaisuyritteen $T \in S_h$ ja mielivaltaisen testifunktion $\Phi_i \in S_h$, saamme

$$(c\partial^t T(t), \Phi_i) = \sum_{j=1}^{M_h} \frac{\partial U_j}{\partial t}(c\Phi_j, \Phi_i) := \sum_{j=1}^{M_h} \mathcal{M}_{ij}\frac{\partial U_j}{\partial t}, \tag{161}$$

missä matriisia $\mathcal{M} \in \mathbb{R}^{M_h \times M_h}$ kutsutaan *globaaliksi massamatriisiksi*. Jakamalla globaali massamatriisi lohkoihin kappaleen 1.3.1 tapaan, päädymme DAE-ryhmään

$$\begin{bmatrix} \mathcal{M}_{FF} & \mathcal{M}_{FI} \\ \mathcal{M}_{IF} & \mathcal{M}_{II} \end{bmatrix} \frac{d}{dt}\begin{bmatrix} \boldsymbol{U}^F \\ \boldsymbol{U}^I \end{bmatrix} + \begin{bmatrix} \mathcal{S}_{FF} & \mathcal{S}_{FI} \\ \mathcal{S}_{IF} & \mathcal{S}_{II} + \mathcal{M}_{II} \end{bmatrix}\begin{bmatrix} \boldsymbol{U}^F \\ \boldsymbol{U}^I \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_F \\ \boldsymbol{g}_I \end{bmatrix}, \tag{162}$$

joka voidaan kirjoittaa tiiviimmin muotoiltuna

$$\big[A\big]\,\dot{\boldsymbol{U}}(t) + \big[B\big]\,\boldsymbol{U}(t) = \boldsymbol{c}(t). \tag{163}$$

2.3.2 Aika-askellus

Haluamme ratkaista kentät eri ajanhetkillä jonolla Δt-välisiä *aika-askeleita*. Merkitään $\boldsymbol{U}^k := \boldsymbol{U}(k\Delta t)$ ja $\boldsymbol{c}^k := \boldsymbol{c}(k\Delta t)$. Soveltamalla Backward Euler-menetelmää yhtälöryhmään (163) päädymme iteraatioon

$$([A] + \Delta t\,[B])\boldsymbol{U}^{k+1} \;=\; \boldsymbol{c}^{k+1}\Delta t + [A]\,\boldsymbol{U}^k, \tag{164}$$

josta voimme ratkaista

$$\boldsymbol{U}^{k+1} \;=\; ([A] + \Delta t\,[B])^{-1}(\boldsymbol{c}^{k+1}\Delta t + [A]\,\boldsymbol{U}^k). \tag{165}$$

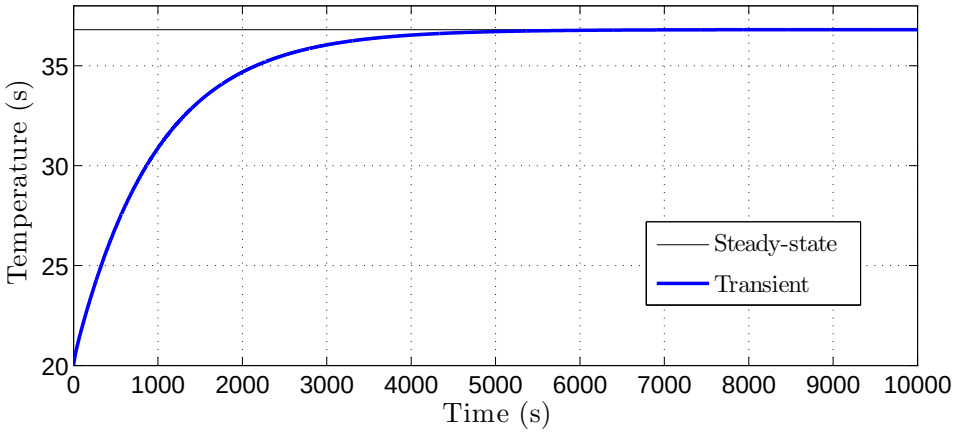
Ensimmäisen asteen kantafunktioita käytetäessä alkuehtoa (157) vastaa oletus

$$\boldsymbol{U}^0 \;=\; \boldsymbol{U}(0) = T_\infty \mathbf{1}, \tag{166}$$

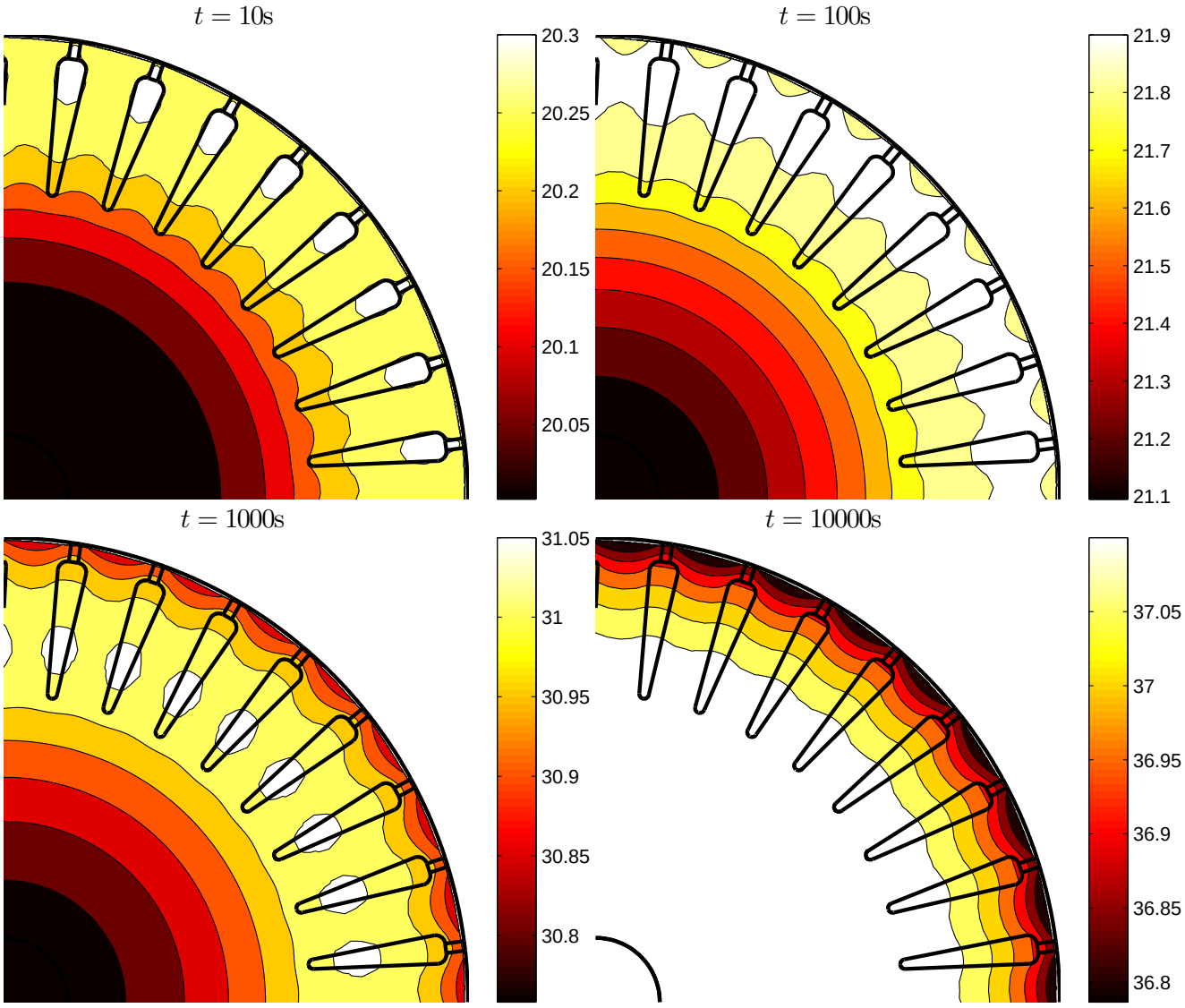
missä $\mathbf{1} = \begin{bmatrix}1 & 1 & \ldots & 1\end{bmatrix}^T \in \mathbb{R}^{M_h}$.

2.3.3 Testiesimerkki

Esimerkkinä parabolisesta tehtävästä ratkaisemme aika-riippuvan version kappaleessa 1.3.3 esitetystä testiesimerkistä. Laskennan toteuttava MATLAB-skripti `robin_transient.m` on esitetty liitteessä A. Kuvassa 2 on esitetty lämpötilan käyttäytyminen ajan funktiona. Kuvasta käy ilmi kuinka lämpötila suppenee kohti steady-state ratkaisua. Kuvassa 3 ratkaistu kenttä T on esitetty eri ajanhetkillä t.



Kuva 2: Pinnan Γ keskilämpötila steady-state ja transienttiratkaisuista.



Kuva 3: Ratkaisut neljällä ajanhetkellä.

A MATLAB-koodi

Toteutuksessa käytetty MATLAB-koodi on pääosin kirjoitettu julkaisujen[3][4] testiesimerkkejä varten. Skriptillä `robin_steadystate.m` suoritettava ratkaisu koostuu seuraavista vaiheista:

1. Luetaan GMsh[5]-ohjelmistolla luotu verkko (`readmesh.m`).

[3] V. Räisänen, S.Suuriniemi, S. Kurz, L. Kettunen, "Subdomain Reduction by Dirichlet-to-Neumann Mappings in Time-Domain Electrical Machine Modeling", *IEEE Transactions on Energy Conversion*, 2015.

[4] V. Räisänen, S.Suuriniemi, S. Kurz, L. Kettunen, "Rapid Computation of Harmonic Eddy-Current Losses in High-Speed Solid-Rotor Induction Machines" *IEEE Transactions on Energy Conversion*, vol. 28, no. 3, pp. 782-790, Sep. 2013.

[5] C. Geuzaine ja J. Remacle. GMSH: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

2. Tehtäväalueelle esitetään materiaaliparametrit ja häviötiheys *regions*-tietorakenteissa. Koodi on suunniteltu pyörrevirtatehtäviä varten, joten jäykkyys- ja massamatriisien materiaaliparametrien nimet ovat vastaavasti `nu` ja `sigma`.

3. Tehtävälle luodaan jäykkyys-matriisi (`stiff_gauss_global.m`).

4. Jäykkyysmatriisi jaetaan lohkoihin vastaten vapausasteiden jakoa (`stiff_partition.m`).

5. Lasketaan häviötiheyttä kuvaava vektori (`forcing_mhquad.m`).

6. Integraalit reunan Γ yli lasketaan jokaiselle reunan kaarelle analyyttisesti.

7. Yhtälöryhmä (106) kootaan ja ratkaistaan.

Transientti-ratkaisu on toteutettu skriptillä `robin_transient.m`, jossa ratkaisuvaiheessa jokaisella aika-askeleella rakennetaan ja ratkaistaan yhtälöryhmä (165).

A.1 Verkkotiedoston lukeminen

Listaus 1: MATLAB-skripti Gmsh-verkkotiedostojen lukemiseen.

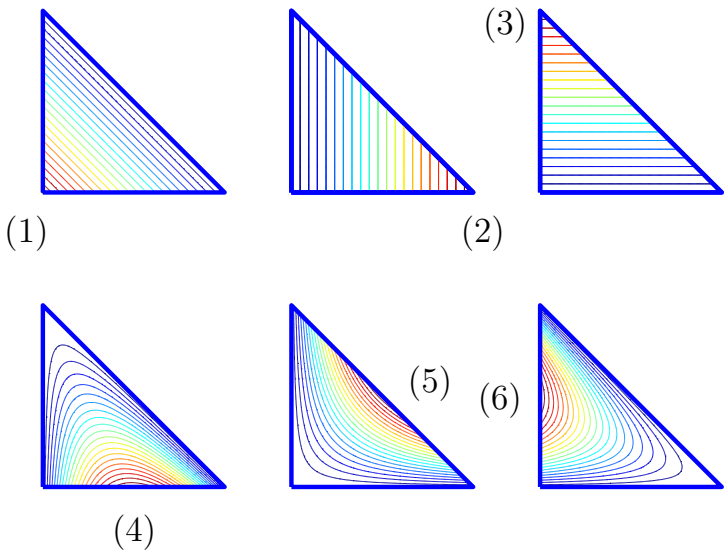
```
% READMESH Read a 2d-GMSH mesh from a .msh-file.
%
% M = READMESH(filename); reads a two-dimensional GMSH-file
% and creates corresponding data structure M.
%
% M.filename        the original filename of the mesh file
% M.num_nodes       number of nodes in the mesh.
% M.num_elements    number of elements in the mesh.
% M.num_lines       number of lines in the mesh
% M.num_triangles   number of triangles in the mesh
%
% M.nodes           M.num_nodes x 4 matrix describing the nodes with
%                   rows [node number, x, y, z];
% M.elements        M.num_elements x 1 cell-array of data structures
%                   with fields describing type, tags and nodes of
%                   each element.
% M.lines           M.num_lines x 1 matrix with indices to each line
%                   in M.elements.
% M.triangles       M.num_triangles x 1 matrix with indices to each
%                   triangle in M.elements.


function M= readmesh(filename)
    M.filename = filename;

    % Before further processing,  read the essential contents of the
    % mesh file.

    fid = fopen(filename, 'r');
    format= readfield(fid);
    nodes = readfield(fid);
    elements = readfield(fid);
    fclose(fid);

    % Read Nodes into a nx4 matrix with rows
    % [node number, x, y, z].
    M.num_nodes = str2num(nodes.lines{1});
    M.num_lines = 0;
    M.num_triangles = 0;
```

```
M.nodes = zeros(M.num_nodes, 4);
if M.num_nodes ~= nodes.linenum-1
    error 'Number of nodes in the mesh mismatch!';
end
for ind_node = 1:nodes.linenum-1
  M.nodes(ind_node, 1:4) = str2num(nodes.lines{ind_node+1});
end

% Read Elements such as lines and triangles.
% .msh-file describes elements as rows
% [number, type, number of tags, tag1, ..., node1, ..],
% where type=1 corresponds to lines and 2 to triangles.

M.num_elements = str2num(elements.lines{1});
M.elements = cell(M.num_elements,1);
if M.num_elements ~= elements.linenum-1
    error 'Number of elements in the mesh mismatch!';
end

for ind_elem = 1:elements.linenum-1
    element= str2num(elements.lines{ind_elem+1});

    elem.items = element;
    elem.nitems = length(element);

    elem.number = element(1);
    elem.type   = element(2);
    elem.ntags  = element(3);

    switch elem.type
        case 1
            M.num_lines = M.num_lines + 1;
        case 2
            M.num_triangles = M.num_triangles + 1;
        case 8
            M.num_lines = M.num_lines + 1;
        case 9
            M.num_triangles = M.num_triangles + 1;
        otherwise
            error 'Unknown element type!'
    end

    % Tags consist of following parameters:
    % 1. The physical entity to which element belongs
    % 2. The elementary geometrical entity to which ..
    % 3. The number of mesh partitions, to which the element belongs.
    % 4- The partition id's

    for ind_tag = 1:elem.ntags
        elem.tags(ind_tag) = element(3+ind_tag);
    end
    elem.physical = elem.tags(1);
    elem.geometrical = elem.tags(2);

    elem.nnodes = elem.nitems-3-elem.ntags;
    for ind_nodes = 1:elem.nnodes;
        elem.nodes(ind_nodes) = element(3+elem.ntags+ind_nodes);
    end
    M.elements{ind_elem} = elem;
end


% Extract lines from the elements

M.lines = zeros(M.num_lines, 1);
ind_line = 0;
```

```
for ind_elem = 1:M.num_elements
    if M.elements{ind_elem}.type == 1 || M.elements{ind_elem}.type == 8
        ind_line = ind_line + 1;
        M.lines(ind_line) = ind_elem;
    end
end

% Extract triangles from the elements
M.triangles = zeros(M.num_triangles, 1);
M.elemtri = zeros(M.num_elements, 1);

ind_triangle = 0;
for ind_elem = 1:M.num_elements
    if M.elements{ind_elem}.type == 2 || M.elements{ind_elem}.type == 9
      ind_triangle = ind_triangle + 1;
        M.triangles(ind_triangle) = ind_elem;
        M.elemtri(ind_elem) = ind_triangle;

        P = M.nodes(M.elements{ind_elem}.nodes, 2:3);
        x1 = P(1, 1); x2 = P(2, 1); x3 = P(3, 1);
        y1 = P(1, 2); y2 = P(2, 2); y3 = P(3, 2);

        M.elements{ind_elem}.grad = [0 1 0; 0 0 1] * ...
                                    inv([1 x1 y1; 1 x2 y2; 1 x3 y3]);
        M.elements{ind_elem}.Je = [x2-x1, x3-x1; y2-y1, y3-y1];
        M.elements{ind_elem}.Jm = 1;%[1 0;0 1];
        M.elements{ind_elem}.vel = 0;
        M.elements{ind_elem}.area = polygon_area(P);
    end
end

M.trianglepoints = zeros(M.num_triangles, 7);
M.trianglephys = zeros(M.num_triangles, 1);
for ind_tri = 1:M.num_triangles
  ind_elem = M.triangles(ind_tri);
  triangle = M.elements{ind_elem};

  M.trianglepoints(ind_tri, 1) = ind_elem;
  M.trianglepoints(ind_tri, 2:3) = M.nodes(triangle.nodes(1), 2:3);
  M.trianglepoints(ind_tri, 4:5) = M.nodes(triangle.nodes(2), 2:3);
  M.trianglepoints(ind_tri, 6:7) = M.nodes(triangle.nodes(3), 2:3);

  M.trianglephys(ind_tri) = triangle.physical;
end
M.trianglepointsx = sortrows(M.trianglepoints, 2);
M.trianglepointsy = sortrows(M.trianglepoints, 3);

M.minx = min(M.nodes(:, 2));
M.maxx = max(M.nodes(:, 2));

M.nodetri  = cell(M.num_nodes, 1);
M.nodephys = cell(M.num_nodes, 1);
for ind_node = 1:M.num_nodes
  M.nodetri{ind_node} = [];
  M.nodephys{ind_node} = [];
end
clear M.physnode;

for ind_tri = 1:M.num_triangles
  tri = M.elements{M.triangles(ind_tri)};
  for ind_node = 1:length(tri.nodes)
    node = tri.nodes(ind_node);
      M.nodetri{node}  = unique([M.nodetri{node} ind_tri]);
      M.nodephys{node} = unique([M.nodephys{node} tri.physical]);
      M.physnode{tri.physical}(node) = 1;
  end
end
```

```
    for ind_line = 1:M.num_lines
      line = M.elements{M.lines(ind_line)};
      for ind_node = 1:length(line.nodes)
        node = line.nodes(ind_node);
          M.nodephys{node} = unique([M.nodephys{node} line.physical]);
      end
    end

    M.nodeorder = zeros(M.num_nodes, 1);
    for ind_node = 1:M.num_nodes
      tri = M.elements{M.triangles(M.nodetri{ind_node}(1))};
      trinode = find(tri.nodes == ind_node);
      if trinode < 4
        M.nodeorder(ind_node) = 1;
      else
        M.nodeorder(ind_node) = 2;
      end
    end

    M.phystriangle = cell(max(M.trianglephys), 1);
    physlist = unique(M.trianglephys);
    for ind_phys = 1:length(physlist)
        phys = physlist(ind_phys);
        M.phystriangle{phys} = find(M.trianglephys == phys);
    end
end

% READFIELD Read a field from GetDP- or Gmsh-file.
%
% F = readfield(fid); reads a field from file identified fid into
% data structure F.
%
% F.name     contains the name of the field
% F.linenum  contains the number of lines of the data in the field.
% F.lines    contains the data of the field.
%
function F= readfield(fid)
    F.name = readline(fid);
    F.linenum = 0;
    while ~feof(fid)
        s = fgets(fid);
        if s(1) == '$'
            break;
        else
            F.linenum = F.linenum+1;
            F.lines{F.linenum} = s;
        end
    end
    s= sprintf('Read␣field␣%s␣with␣%d␣lines', F.name, F.linenum);
  %  disp(s);
end

% Remove '\n'-character from the end of a string
function s = readline(fid)
    s = fgets(fid);
    s = s(1:(length(s)-1));
end

function A = polygon_area(P)

[nVertices, tmp] = size(P);

u = P(:, 1) + [P(2:nVertices, 1);P(1, 1)];
v = [P(2:nVertices, 2);P(1, 2)] - P(:, 2);

A = abs(0.5*u'*v);

end
```

A.2 Numeerinen integrointi



Kuva 4: Kantafunktiot referenssielementillä. Robin-tehtävän implementaatiossa käytetään ainoastaan ensimmäisen asteen kantafunktioita (1)-(3).

Listaus 2: MATLAB-skripti kantafunktioiden arvojen laskentaan referenssielementillä.

```
function val = bf_node(ind, u, v)

% The order here is according to 2nd order node ordering for GMsh
switch(ind)
  case 1
    val = 1-u-v;
  case 2
    val = u;
  case 3
    val = v;
  case 4
    val = (1-u-v)*u;
  case 5
    val = u*v;
  case 6
    val = (1-u-v)*v;
  otherwise,
    error 'invalid node index'
end
```

Listaus 3: MATLAB-skripti kantafunktioiden gradienttien arvojen laskentaan referenssielementillä.

```
function val = bf_gradnode(ind, u, v)

switch(ind)
```

```
  case 1
    val = [-1;-1];
  case 2
    val = [1;0];
  case 3
    val = [0;1];
  case 4
    val = [1-v-2*u; -u];
  case 5
    val = [v; u];
  case 6
    val = [-v; 1-u-2*v];
  otherwise,
    error 'invalid node index'
end
```

Listaus 4: MATLAB-koodi jäykkyys- ja massamatriisien rakentamiseen.

```
function [Sg, Mg] = stiff_global_gauss(M, regions, order, bforder)

gaussdata;
if nargin < 4
  bforder = 2;
end
if nargin < 3
  order = 6;
end

G = GAUSS{order};
X = G(:, 1); Y = G(:, 2); W = G(:, 3);

% Number of gauss points
npoints = length(W);
nfun = 3 * bforder;

CSe = cell(M.num_triangles, 1);
CMe = cell(M.num_triangles, 1);
SgL = zeros(M.num_triangles*(nfun^2), 3);
MgL = zeros(M.num_triangles*(nfun^2), 3);
ind_l = 0;

% Organize the values of the basis functions and their gradients into
% an matrix/array.
BF = zeros(nfun, npoints);
BFg = zeros(2, nfun, npoints);

for ind_BF = 1:nfun
  for ind_point = 1:npoints
    u = X(ind_point);
    v = Y(ind_point);
    BF(ind_BF, ind_point) = bf_node(ind_BF, u, v);
    BFg(:, ind_BF, ind_point) = bf_gradnode(ind_BF, u, v);
  end
end

for ind_triangle = 1:M.num_triangles
  tri = M.elements{M.triangles(ind_triangle)};
  nnodes = length(tri.nodes);

  % Corner points of the triangle tri.
  P = M.nodes(tri.nodes, 2:3);
  x1 = P(1, 1); y1 = P(1, 2);
  x2 = P(2, 1); y2 = P(2, 2);
  x3 = P(3, 1); y3 = P(3, 2);

  sigma = regions{tri.physical}.sigma;
  if isfield(tri, 'nu')
```

```
      nu = tri.nu;
  else
    nu = regions{tri.physical}.nu;
    if length(nu) == 1
        nu = nu*ones(npoints, 1);
    end
  end

  % The Jacobian term for the transformation of the gradients.
  Jterm = inv(tri.Je)*inv(tri.Je');
  invJ = inv(tri.Je);

  % Element stiffness- and mass matrices
  Se = zeros(3 * bforder);
  Me = zeros(3 * bforder);

  % Compute element stiffnes- and mass matrices.
  for ind_point = 1:npoints
    FVALUE = BF(:, ind_point)*BF(:, ind_point)';
    FGRADVALUE = BFg(:, :, ind_point)' * Jterm * BFg(:, :, ind_point);

    Me = Me + abs(det(tri.Je)) * W(ind_point) * sigma * FVALUE;
    Se = Se + abs(det(tri.Je)) * W(ind_point) * nu(ind_point) * FGRADVALUE;
  end
  CSe{ind_triangle} = Se;
  CMe{ind_triangle} = Me;

  % Assemble list of contributions to the elements of the global stiffness
  % and mass matrices.
  for ind_row = 1:nfun
        globrow = tri.nodes(ind_row);
    for ind_col = 1:nfun
        ind_l = ind_l + 1;
          globcol = tri.nodes(ind_col);
          SgL(ind_l, 1:3) = [globrow, globcol, Se(ind_row, ind_col)];
          MgL(ind_l, 1:3) = [globrow, globcol, Me(ind_row, ind_col)];
    end
  end
end

SgL(find(SgL(:, 1)==0), :) = [];
MgL(find(MgL(:, 1)==0), :) = [];

% Convert the lists to a sparse matrices.
Sg = sparse(SgL(:, 1), SgL(:, 2), SgL(:, 3), M.num_nodes, M.num_nodes);
Mg = sparse(MgL(:, 1), MgL(:, 2), MgL(:, 3), M.num_nodes, M.num_nodes);
```

Listaus 5: MATLAB-koodi jäykkyys- ja massamatriisien jakamiseen lohkoihin.

```
function [SC, MC, part, freenodes, inds, permfull] = ...
          stiff_partition(M, part, Sg, Mg)

num_partitions = length(part);

% Divide nodes of the mesh M into partitions according to physical list.
% If two partitions are associated to a given node, the node is given to
% the partition with higher priority. If two partitions associated to
% a node have equal priority, the node is made free. This is appropriate
% for the node in the intersection of two symmetry lines.

nodespart = zeros(M.num_nodes, 2);
physlistt = [];
for ind_node = 1:M.num_nodes
    nodephys = M.nodephys{ind_node};

    for ind_part = 1:num_partitions
        physlist = part{ind_part}.physlist;
```

```
        if ~isempty(intersect(physlist, nodephys))
            if part{ind_part}.priority > nodespart(ind_node, 2)
                nodespart(ind_node, :) = [ind_part, part{ind_part}.priority];
            elseif part{ind_part}.priority == nodespart(ind_node, 2)
                nodespart(ind_node, :) = [0 0];
            end
        end
    end
end

% Initialize the permutation matrix used for rearraging the degrees of
% freedom.
permfull = sparse(M.num_nodes, M.num_nodes);

% Free nodes are the nodes not associated to any Physical Line or Physical
% Surface. They are given the lowest indices. The remaining degrees of
% freedom are first sorted according to partition number. Nodes within
% each partition are sorted according to the rule given in part{}.sorting.

freenodes = find(nodespart(:, 1) == 0);
for ind_node = 1:length(freenodes)
    permfull(ind_node, freenodes(ind_node)) = 1;
end

ind0 = length(freenodes);

% The array inds contains the first and last DoF number of each partition.
inds = zeros(num_partitions + 1, 2);
inds(1, 1) = 1;
inds(1, 2) = ind0;

ind0 = ind0 + 1;

for ind_part = 1:num_partitions
    part{ind_part}.nodes = find(nodespart(:, 1) == ind_part);
    part{ind_part}.num_nodes = length(part{ind_part}.nodes);

    nodes = part{ind_part}.nodes;
    nodes = [nodes, 0*nodes, 0*nodes];
    nodes(:, 3) = M.nodeorder(nodes(:, 1)) * part{ind_part}.sort_BForder;

    p = M.nodes(nodes(:, 1), 2:3);
    p(:, 1) = p(:, 1) - part{ind_part}.sort_origin(1);
    p(:, 2) = p(:, 2) - part{ind_part}.sort_origin(2);

    switch part{ind_part}.sorting
        case 1 % Angle
            nodes(:, 2) = atan2(p(:, 2), p(:, 1));
        case 2 % Radial
            nodes(:, 2) = abs(p(:, 1).^2 + p(:, 2).^2);
        case 3
            nodes(:, 2) = p(:, 1);
        case 4
            nodes(:, 2) = p(:, 2);
        otherwise
    end
    nodes = sortrows(nodes, [3 2]);
    part{ind_part}.permlist = nodes(:, 1);

    for ind_node = 1:part{ind_part}.num_nodes
        permfull(ind0 + ind_node - 1, nodes(ind_node, 1)) = 1;
    end

    part{ind_part}.nodes = nodes;
    ind1 = ind0 + part{ind_part}.num_nodes;
    inds(ind_part + 1, :) = [ind0, ind1-1];
    ind0 = ind1;
```

```
end

% Construct the sorted global stiffness and mass matrices.

Sg_resorted = permfull*Sg*permfull';
Mg_resorted = permfull*Mg*permfull';

% Divide the new global matrices into cell arrays.

SC = cell(num_partitions + 1);
MC = cell(num_partitions + 1);

for ind_partr = 1:num_partitions+1
    for ind_partc = 1:num_partitions+1
        SC{ind_partr, ind_partc} = ...
        Sg_resorted(inds(ind_partr, 1):inds(ind_partr, 2), ...
                    inds(ind_partc, 1):inds(ind_partc, 2));
        MC{ind_partr, ind_partc} = ...
        Mg_resorted(inds(ind_partr, 1):inds(ind_partr, 2), ...
                    inds(ind_partc, 1):inds(ind_partc, 2));
    end
end
```

Listaus 6: MATLAB-skripti häviövektorin laskemiseen.

```
function [F, ab] = forcing_mhquad(M, MasterNodes, regions, num_eta)

nmaster = length(MasterNodes);
ff = zeros(nmaster, num_eta);

for ind_master = 1:nmaster
     nodetri = M.nodetri{MasterNodes(ind_master)};

    for ind_tri = 1:length(nodetri)
      tri = M.elements{M.triangles(nodetri(ind_tri))};
      ind_node = find(tri.nodes == MasterNodes(ind_master));
      if length(ind_node) ~= 1
        error 'Node not found!';
      end
      if ind_node < 4
        ff(ind_master, :) = ff(ind_master, :) ...
                          + tri.area * regions{tri.physical}.Js / 3;
      else
        ff(ind_master, :) = ff(ind_master, :) ...
                          + tri.area * regions{tri.physical}.Js / 12;
      end
    end
end
F =  ff;
```

A.3 Steady-state Robin-tehtävän toteutus

Listaus 7: MATLAB-skripti steady-state Robin-tehtävän ratkaisemiseen.

```
tic
r0 = 0.0725;                    % Outer radius of the rotor
Tenv = 20;                      % Temperature of the environment
meshfile = 'arkkio_rotor.msh';  % Filename of the mesh file
num_gauss = 12;                 % Number of gauss points
bforder = 1;                    % Order of the basis functions.

% To obtain semi-reasonable results, a coefficient of convection was
% taken from:
% Howey, Childs, Holmes -  Air-Gap Convection in Rotating Electrical Machines
```

```
hconvection = 130;                % W/Km^2

% Analytical solution can be obtained easily when the whole rotor is given
% homogenous thermal conductivity and loss density.
analytical_test = false;

% Data structures related to the rotor slots are stored in data file
% regiondata generated by code/arkkio_rotor.m
load regiondata;

if ~exist('M')
    disp 'Loading mesh file'
    toc
    M = readmesh(meshfile);
end

slotarea = mesh_physarea(M, R.phys_wedges(1)) ...
         + mesh_physarea(M, R.phys_slots(1));
% At one percent slip, the loss per unit length is approximately 1kW/m.
% Thus, the loss density per unit length
Ptot = 1000;
peddy = (Ptot/(R.Q*slotarea));

% https://en.wikipedia.org/wiki/Heat_capacity                  (Iron)
% https://en.wikipedia.org/wiki/List_of_thermal_conductivities (Steel)
regions{R.phys_coresurface}.nu = 50;
regions{R.phys_coresurface}.sigma = 3.54e6;
regions{R.phys_coresurface}.type = 1;
regions{R.phys_coresurface}.Js = 0;
regions{R.phys_coresurface}.nl = false;

% https://en.wikipedia.org/wiki/Heat_capacity                  (Aluminium)
% https://en.wikipedia.org/wiki/List_of_thermal_conductivities (Aluminium)
clear reg_rbar
reg_rbar.nu = 204;
reg_rbar.sigma = 2.42e6;
reg_rbar.type = 1;
reg_rbar.Js = peddy;
reg_rbar.nl = false;

if analytical_test
    regions{R.phys_coresurface} = reg_rbar;
end

reg_wedge = reg_rbar;
for ind_bar = 1:R.Q
    regions{R.phys_slots(ind_bar)} = reg_rbar;
    regions{R.phys_wedges(ind_bar)} = reg_wedge;
end

regions{R.phys_innerboundary}.type = 3;   % Gamma

disp 'Building the global stiffness and mass matrices.'
toc
if ~exist('Sg')
  [Sg, Mg] = stiff_global_gauss(M, regions, num_gauss, bforder);
end

disp 'Sorting the degrees of freedom.'
toc
if ~exist('SC')
    part = cell(1, 1);
    part{1}.name     = 'Rotor-Airgap Interface';
    part{1}.type     = 1;
    part{1}.physlist = R.phys_innerboundary;
    part{1}.priority = 1;
    part{1}.sorting  = 1;
    part{1}.sort_origin  = [0, 0];
```

```
    part{1}.sort_BForder = false;
    part{1}.sort_invert  = false;

    [SC,MC,part,freenodes,inds,permfull] = stiff_partition(M, part, Sg, Mg);

    S_FF = SC{1, 1}; S_FI = SC{1, 2};
    S_IF = SC{2, 1}; S_II = SC{2, 2};
    M_FF = MC{1, 1}; M_FI = MC{1, 2};
    M_IF = MC{2, 1}; M_II = MC{2, 2};
end

numF = length(SC{1, 1}); % The number of free nodes outside Gamma.
numI = length(SC{2, 2}); % The number of free nodes on Gamma.
nfun = 3*bforder;        % The number of nodes on each triangle.

disp 'Computing forcing vector.'
toc
F = forcing_mhquad(M, freenodes, regions, 1);

disp 'Computing line integrals.'
toc
G  = sparse(numI, numI);
G2 = sparse(numI, numI);
for ind_node = 1:numI
    ind_prev = mod(ind_node-2, numI)+1;
    ind_next = mod(ind_node, numI)+1;

    node_prev = part{1}.nodes(ind_prev, 1);
    node      = part{1}.nodes(ind_node, 1);
    node_next = part{1}.nodes(ind_next, 1);

    len_prev= norm(M.nodes(node_prev, 2:3) - M.nodes(node, 2:3))
    len_next= norm(M.nodes(node_next, 2:3) - M.nodes(node, 2:3));

    G(ind_node, ind_node) = hconvection*(len_prev/3 + len_next/3);
    G(ind_node, ind_next) = hconvection*(len_next)/6;
    G(ind_node, ind_prev) = hconvection*(len_prev)/6;
end

disp 'Assembly and solution.'
toc
T = [S_FF, S_FI; S_IF  S_II+G]\[F;G*ones(numI, 1)*Tenv];
%T = S_FF\F;

% Values on the interface Gamma
Tref0 = Tenv + Ptot/(2*pi*r0*hconvection);
Tgamma = T(numF + (1:numI));
disp 'Post-processing:.'
toc
disp ' '
disp 'Global:'
disp(sprintf('Min/Max Temperature       : %.3fC / %.3fC', ...
             min(T), max(T)));
disp ' '
disp 'Interface:'
disp(sprintf('Min/Avg/Max Interface Temp: %.3fC / %.3fC / %.3fC', ...
             min(Tgamma), sum(Tgamma)/numI, max(Tgamma)));
disp(sprintf('Reference Temperature     : %.3fC', Tref0));
```

A.4 Transientin Robin-tehtävän toteutus

Listaus 8: MATLAB-skripti transientti Robin-tehtävän ratkaisemiseen.

```
tic
r0 = 0.0725;                    % Outer radius of the rotor
```

```
Tenv = 20;                          % Temperature of the environment
meshfile = 'arkkio_rotor.msh';      % Filename of the mesh file
num_gauss = 12;                     % Number of gauss points
bforder = 1;                        % Order of the basis functions.
dt = 1;                             % Time step size (in seconds)
num_timesteps = 10000;              % Number of time steps
visualize = false;                  % Draw results during computation

% To obtain semi-reasonable results, a coefficient of convection was
% taken from:
% Howey, Childs, Holmes -  Air-Gap Convection in Rotating Electrical Machines
hconvection = 130;                  % W/Km^2

% Analytical solution can be obtained easily when the whole rotor is given
% homogenous thermal conductivity and loss density.
analytical_test = false;

% Data structures related to the rotor slots are stored in data file
% regiondata generated by code/arkkio_rotor.m
load regiondata;

if ~exist('M')
    disp 'Loading mesh file'
    toc
    M = readmesh(meshfile);
end

slotarea = mesh_physarea(M, R.phys_wedges(1)) ...
         + mesh_physarea(M, R.phys_slots(1));
% At one percent slip, the loss per unit length is approximately 1kW/m.
% Thus, the loss density per unit length
Ptot = 1000;
peddy = (Ptot/(R.Q*slotarea));

% https://en.wikipedia.org/wiki/Heat_capacity                  (Iron)
% https://en.wikipedia.org/wiki/List_of_thermal_conductivities (Steel)
regions{R.phys_coresurface}.nu = 50;
regions{R.phys_coresurface}.sigma = 3.54e6;
regions{R.phys_coresurface}.type = 1;
regions{R.phys_coresurface}.Js = 0;
regions{R.phys_coresurface}.nl = false;

% https://en.wikipedia.org/wiki/Heat_capacity                  (Aluminium)
% https://en.wikipedia.org/wiki/List_of_thermal_conductivities (Aluminium)
clear reg_rbar
reg_rbar.nu = 204;
reg_rbar.sigma = 2.42e6;
reg_rbar.type = 1;
reg_rbar.Js = peddy;
reg_rbar.nl = false;

if analytical_test
    regions{R.phys_coresurface} = reg_rbar;
end

reg_wedge = reg_rbar;
for ind_bar = 1:R.Q
    regions{R.phys_slots(ind_bar)} = reg_rbar;
    regions{R.phys_wedges(ind_bar)} = reg_wedge;
end

regions{R.phys_innerboundary}.type = 3;   % Gamma

disp 'Building the global stiffness and mass matrices.'
toc
if ~exist('Sg')
  [Sg, Mg] = stiff_global_gauss(M, regions, num_gauss, bforder);
end
```

```
disp 'Sorting the degrees of freedom.'
toc
if ~exist('SC')
    part = cell(1, 1);
    part{1}.name      = 'Rotor-Airgap Interface';
    part{1}.type      = 1;
    part{1}.physlist  = R.phys_innerboundary;
    part{1}.priority  = 1;
    part{1}.sorting   = 1;
    part{1}.sort_origin   = [0, 0];
    part{1}.sort_BForder = false;
    part{1}.sort_invert   = false;

    [SC,MC,part,freenodes,inds,permfull] = stiff_partition(M, part, Sg, Mg);

    S_FF = SC{1, 1}; S_FI = SC{1, 2};
    S_IF = SC{2, 1}; S_II = SC{2, 2};
    M_FF = MC{1, 1}; M_FI = MC{1, 2};
    M_IF = MC{2, 1}; M_II = MC{2, 2};
end

numF = length(SC{1, 1}); % The number of free nodes outside Gamma.
numI = length(SC{2, 2}); % The number of free nodes on Gamma.
nfun = 3*bforder;        % The number of nodes on each triangle.

disp 'Computing forcing vector.'
toc
F = forcing_mhquad(M, freenodes, regions, 1);

disp 'Computing line integrals.'
toc
G  = sparse(numI, numI);
G2 = sparse(numI, numI);
for ind_node = 1:numI
    ind_prev = mod(ind_node-2, numI)+1;
    ind_next = mod(ind_node, numI)+1;

    node_prev = part{1}.nodes(ind_prev, 1);
    node      = part{1}.nodes(ind_node, 1);
    node_next = part{1}.nodes(ind_next, 1);

    len_prev= norm(M.nodes(node_prev, 2:3) - M.nodes(node, 2:3))
    len_next= norm(M.nodes(node_next, 2:3) - M.nodes(node, 2:3));

    G(ind_node, ind_node) = hconvection*(len_prev/3 + len_next/3);
    G(ind_node, ind_next) = hconvection*(len_next)/6;
    G(ind_node, ind_prev) = hconvection*(len_prev)/6;
end

disp 'Assembly and solution.'
toc
TT = zeros(numF + numI, num_timesteps);
T0 = ones(numF + numI, 1)*Tenv;
Tmin = zeros(num_timesteps, 2);
Tavg = zeros(num_timesteps, 2);
Tmax = zeros(num_timesteps, 2);

for timestep = 1:num_timesteps
    A = [M_FF, M_FI; M_IF, M_II];
    B = [S_FF, S_FI; S_IF S_II+G];
    c = [F;G*ones(numI, 1)*Tenv];

    T = (A + dt*B)\(dt*c + A*T0);

    TT(:, timestep) = T;
    Tgamma = T(numF + (1:numI));
```

```
    Tmin(timestep, 1) = min(T);
    Tavg(timestep, 1) = sum(T)/(numI+numF);
    Tmax(timestep, 1) = max(T);

    Tmin(timestep, 2) = min(Tgamma);
    Tavg(timestep, 2) = sum(Tgamma)/numI;
    Tmax(timestep, 2) = max(Tgamma);

    % Values on the interface Gamma
    Tgamma = T(numF + (1:numI));
    disp(sprintf('\nTimestep %d/%d', timestep, ...
        num_timesteps));
    disp(sprintf('Global Min/Max Temperature        : %.3fC / %.3fC', ...
                 min(T), max(T)));
    disp(sprintf('Min/Avg/Max Interface Temp: %.3fC / %.3fC / %.3fC', ...
                  min(Tgamma), sum(Tgamma)/numI, max(Tgamma)));

    T0 = T;
    if mod(timestep, 200) == 0 && visualize
        figure(1);
        drawsolution(T, M, freenodes);
        drawnow;
        colormap hot

        figure(2);
        clf
        hold on
        plot(dt*(1:timestep), Tmin(1:timestep, 1), 'b', 'LineWidth', 2);
        plot(dt*(1:timestep), Tavg(1:timestep, 1), 'g', 'LineWidth', 2);
        plot(dt*(1:timestep), Tmax(1:timestep, 1), 'r', 'LineWidth', 2);
        plot(dt*(1:timestep), Tmin(1:timestep, 2), 'b--', 'LineWidth', 2);
        plot(dt*(1:timestep), Tavg(1:timestep, 2), 'g--', 'LineWidth', 2);
        plot(dt*(1:timestep), Tmax(1:timestep, 2), 'r--', 'LineWidth', 2);
        xlim([0, num_timesteps]);
        xlabel 'Time (s)'
        ylabel 'Temperature (C)'
        drawnow;
    end
end


% Values on the interface Gamma
Tgamma = T(numF + (1:numI));
disp 'Post-processing:.'
toc
disp ' '
disp 'Global:'
disp(sprintf('Min/Max Temperature        : %.3fC / %.3fC', ...
             min(T), max(T)));
disp ' '
disp 'Interface:'
disp(sprintf('Min/Avg/Max Interface Temp: %.3fC / %.3fC / %.3fC', ...
             min(Tgamma), sum(Tgamma)/numI, max(Tgamma)));
disp(sprintf('Reference Temperature      : %.3fC', Tref0));
```