ESF EIROPAS SOCIĀLAIS FONDS IEGULDĪJUMS TAVĀ NĀKOTNĒ

29.11.2012
Raimonds Viļums (raimonds.vilums@lu.lv)
LU Fizikas un matemātikas fakultāte

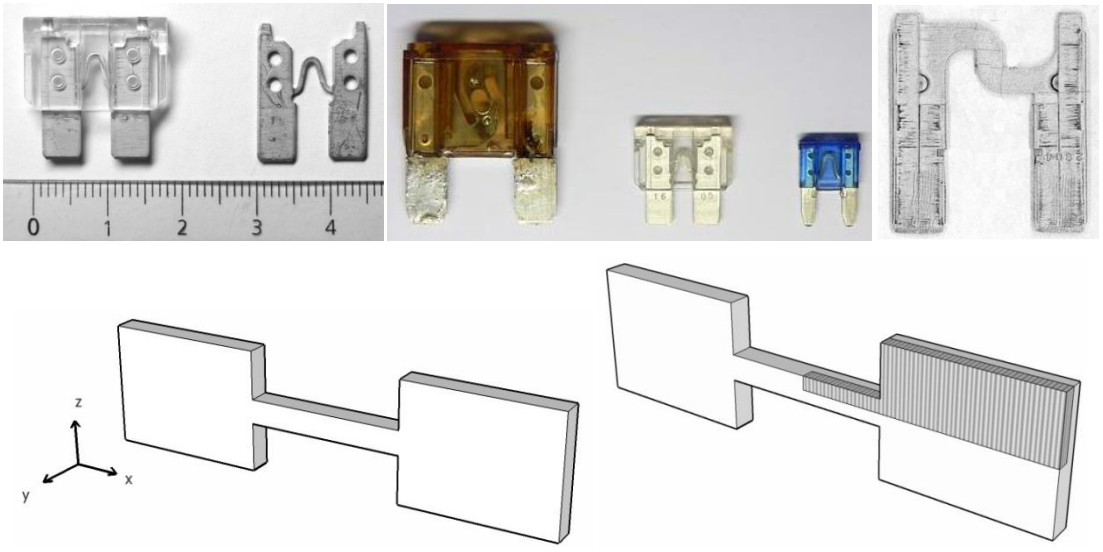
II. Auto drošinātājs

Matemātiskā modelēšana, izmantojot OpenFOAM

Šajā rakstā apskatīsim automašīnu elektriskā drošinātāja modelēšanu, izmantojot datorprogrammu un C++ bibliotēku pakotni OpenFOAM. Iepriekšējais dokuments par temperatūru elektriskajā vadā[1] satur daudzus sīkākus paskaidrojumus piemēra sagatavošanā un rēķināšanā, kas šeit netiks atkārtoti. Pamācība veidota, lietojot OpenFOAM 2.1.x ar atbilstošo *swak4foam* versiju, kā arī *ParaView* 3.10.

1. Fizikālā situācija un problēmas formulējums Dekarta koordinātu sistēmā

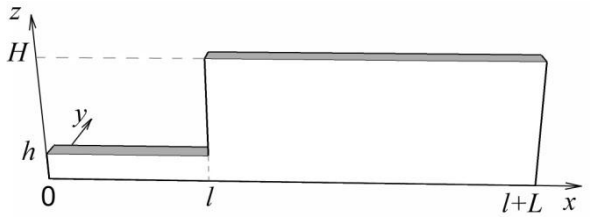
Dots automašīnas drošinātājs, kura ģeometriskais modelis veidots no paralēlskaldņiem. Uzdevums ir noskaidrot, cik ilgs laiks paiet līdz metāla kušanai pēc strāvas padeves uzsākšanas, kā arī – kāds ir temperatūras sadalījums pašā drošinātājā. Piemērs ņemts no promocijas darba.[2]



[1] Viļums R. *I. Temperatūra elektriskajā vadā un tā izolācijā : Matemātiskā modelēšana, izmantojot OpenFOAM.* Rīga : LU FMF, 2012. Pieejams: http://matmod.pbworks.com/w/file/59999645/1-ElektriskaisVads.pdf

[2] Viļums R. *Conservative Averaging Method in Mathematical Models of Heat Processes of Electric Systems*. Riga : University of Latvia, 2010. PhD Thesis.

Simetrijas dēļ ņemam astoto daļu no pilnās ģeometrijas:



Ja temperatūru apzīmē ar funkciju $T(x,t)$, tad sekojošs Laplasa vienādojums raksturo siltuma plūsmu materiālā:

$$\frac{\partial c\rho T}{\partial t} - \nabla \cdot k\nabla T = F, \tag{1}$$

kur k ir īpatnējais siltumvadīšanas, bet $\gamma = c\rho$ siltumietilpības (pēc tilpuma) koeficients. Fizikas rokasgrāmatās šo parametru atkarība no temperatūras parasti dota tabulas veidā, ko mēs aproksimēsim ar trešās pakāpes polinomiem. F ir lineāra avota funkcija, kas balstīta uz Džoula likumu un raksturo līdzstrāvas rezultātā radušos siltumu:

$$F_i(T) = \frac{I^2}{A_i^2}\rho_{ref}\left(1-\alpha_{ref}\left(T-T_{ref}\right)\right), \qquad i \in \{0,1\}, \tag{2}$$

kur indekss $i=0$ norāda uz tievāko apgabalu, bet $i=1$ – uz lielāko; I – strāvas stiprums, A – šķērsgriezuma laukums, ρ_{ref} – pretestība references jeb atsauces temperatūrā, α_{ref} – temperatūras lineārais koeficients pretestībai, T_{ref} – references temperatūra.

Uz ārējām virsmām norādām robežnosacījumus, kas raksturo siltumapmaiņu ar apkārtējo vidi:

$$\left(k\frac{\partial T}{\partial y} + \alpha\left(T-T_\infty\right)\right)\Big|_{y=b} = 0, \tag{3}$$

$$\left(k\frac{\partial T}{\partial z} + \alpha\left(T-T_\infty\right)\right)\Big|_{z=h,x\in[0,l]} = 0, \qquad \left(k\frac{\partial T}{\partial z} + \alpha\left(T-T_\infty\right)\right)\Big|_{z=H,x\in[l,l+L]} = 0, \tag{4}$$

$$\left(-k\frac{\partial T}{\partial x} + \alpha\left(T-T_\infty\right)\right)\Big|_{x=l,z\in[h,H]} = 0, \tag{5}$$

kur α ir virsmas siltumapmaiņas koeficients un T_∞ – apkārtējās vides temperatūra.

Pārējās ir simetrijas virsmas:

$$\frac{\partial T}{\partial x}\Big|_{x=0} = 0, \quad \frac{\partial T}{\partial x}\Big|_{x=l+L} = 0, \tag{6}$$

$$\frac{\partial T}{\partial y}\Big|_{y=0} = 0, \quad \frac{\partial T}{\partial z}\Big|_{z=0} = 0. \tag{7}$$

2. Risinātāja pirmkods

Programmu nosaucam par *jouleHeatingFoam*. Pirmkods veidots pēc *laplacianFoam* un iepriekšējā rakstā izveidotā *laplacianSourceFoam* parauga un satur sekojošas datnes ar attēlā redzamo struktūru.

jouleHeatingFoam.C

```
#include "fvCFD.H"
#include "simpleControl.H"

int main(int argc, char *argv[])
{

#include "setRootCase.H"
#include "createTime.H"
#include "createMesh.H"
#include "createFields.H"

simpleControl simple(mesh)

Info<< "\nCalculating temperature distribution\n" << endl;

while (simple.loop())
{
   Info<< "Time = " << runTime.timeName() << nl << endl;

   while (simple.correctNonOrthogonal())
   {
      //siltumietilpība, siltumvadāmība un avota funkcija
      CRho = CRho0 + CRho1*T + CRho2*sqr(T) + CRho3*pow(T,3);
      K    = K0    + K1*T    + K2*sqr(T)    + K3*pow(T,3);
      F    = sqr(I) * rhoRef / sqr(A) * (1+alphaRef*(T-TRef));

      solve
      (
          //diferenciālvienādojums (1)
          fvm::ddt(CRho, T)
        - fvm::laplacian(K, T)
          ==
          F
       );
   }

   runTime.write();

   Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
       << "  ClockTime = " << runTime.elapsedClockTime() << " s"
       << nl << endl;
}

Info<< "End\n" << endl;

return 0;
}
```

createFields.H

```
    Info<< "Reading fields: T, CRho, K, A\n" << endl;

    //Temperatūra
    volScalarField T
    (
        IOobject
        (
            "T",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    //Definējam siltumvadītspējas un siltumietilpības fizikālās dimensijas
    dimensionSet dimHeatConductivity (dimPower/dimLength/dimTemperature);
    dimensionSet dimHeatCapacity (dimSpecificHeatCapacity * dimDensity);

    //Siltumvadīšanas koeficients
    volScalarField K
    (
        IOobject
        (
            "K",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatConductivity, 0.0)
    );

    volScalarField K0
    (
        IOobject
        (
            "K0",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        mesh
    );

    volScalarField K1
    (
        IOobject
        (
            "K1",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatConductivity/dimTemperature, 0.0)
    );

    volScalarField K2
    (
        IOobject
```

```
        (
            "K2",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("z", dimHeatConductivity/pow(dimTemperature,2), 0)
    );

    volScalarField K3
    (
        IOobject
        (
            "K3",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("z", dimHeatConductivity/pow(dimTemperature,3), 0)
    );

    //Siltumietilpības koeficients
    volScalarField CRho
    (
        IOobject
        (
            "CRho",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatCapacity, 0)
    );

    volScalarField CRho0
    (
        IOobject
        (
            "CRho0",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        mesh
    );

    volScalarField CRho1
    (
        IOobject
        (
            "CRho1",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatCapacity/dimTemperature, 0)
    );
```

```
    volScalarField CRho2
    (
        IOobject
        (
            "CRho2",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatCapacity/pow(dimTemperature,2), 0)
    );

    volScalarField CRho3
    (
        IOobject
        (
            "CRho3",
            runTime.constant(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimHeatCapacity/pow(dimTemperature,3), 0)
    );

    //Šķērsgriezuma laukums
    volScalarField A
    (
        IOobject
        (
            "A",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        mesh
    );

    //Avota funkcija
    volScalarField F
    (
        IOobject
        (
            "F",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh,
        dimensionedScalar("zero", dimPower/dimVolume, 0.0)
    );

    //Nolasām pārējos parametrus no "vārdnīcas", ko izmantos formula (2)
    Info<< "Reading electricProperties\n" << endl;

    IOdictionary electricProperties
    (
        IOobject
        (
            "electricProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ,
```

```
            IOobject::NO_WRITE
        )
    );

    dimensionedScalar rhoRef
    (
        electricProperties.lookup("rhoRef")
    );
    dimensionedScalar alphaRef
    (
        electricProperties.lookup("alphaRef")
    );
    dimensionedScalar TRef
    (
        electricProperties.lookup("TRef")
    );
    dimensionedScalar I
    (
        electricProperties.lookup("I")
    );
```

Make/files

```
jouleHeatingFoam.C

EXE = $(FOAM_USER_APPBIN)/jouleHeatingFoam
```

Make/options

```
EXE_INC = -I$(LIB_SRC)/finiteVolume/lnInclude

EXE_LIBS = -lfiniteVolume
```

Kompilējam programmu, izpildot *wmake*.

```
wmake
```

Veiksmīgas kompilācijas gadījumā direktorijā *$FOAM_USER_APPBIN* parādīsies izpildāmā datne *jouleHeatingFoam*.

3. Piemēra datņu sagatavošana

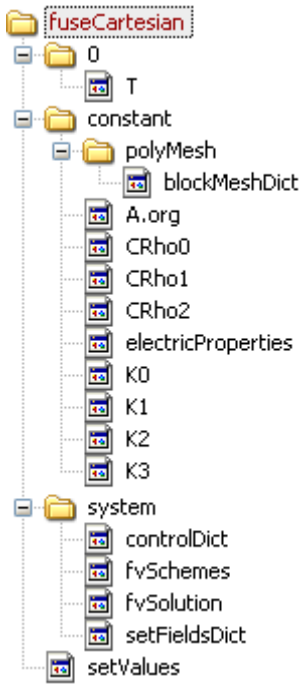
Piemēru nosaucam par *fuseCartesian*, kam datņu struktūra būs kā tālāk redzamajā attēlā. Par paraugu ņemam cinka drošinātāju ar 50 ampēru nominālvērtību, kas apskatīts minētā promocijas darba 3.7. nodaļā, un kam ir sekojoši izmēri un fizikālās īpašības:

$$l = 13mm,\quad L = 27mm,\quad h = 1.9mm,\quad H = 8mm,\quad b = 0.2mm, \tag{8}$$

$$k_{Zn} \simeq 119.477 - 0.0230790T - 3.84199\cdot 10^{-5}T^2 - 4.05844\cdot 10^{-8}T^3, \tag{9}$$

$$\gamma_{Zn} \simeq 2.745050\cdot 10^6 + 1087.325T + 0.445625T^2, \tag{10}$$

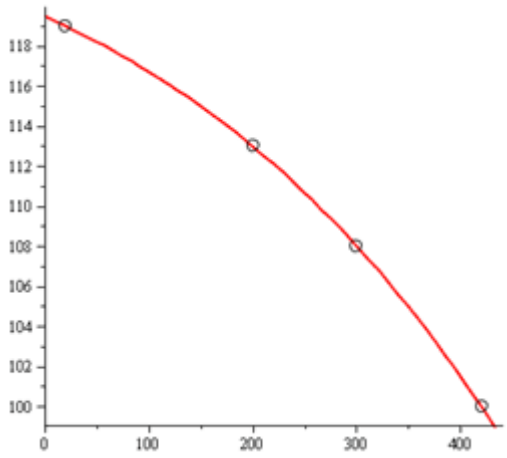
$$\rho_{ref} = 5.95\cdot 10^{-8},\quad \alpha_{ref} = 4.20\cdot 10^{-3},\quad T_{ref} = 20°\mathrm{C}. \tag{11}$$



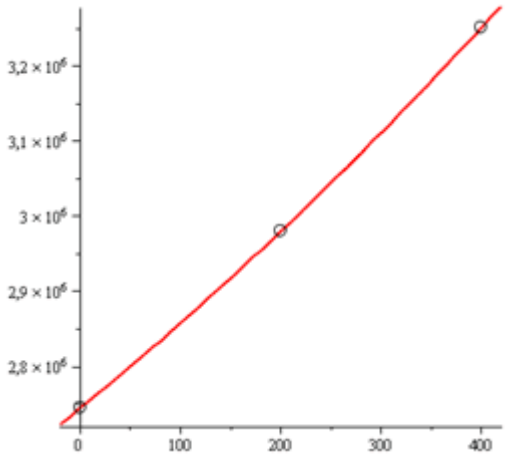
Interpolācijas polinomi (9) un (10) iegūti ar programmas Maple palīdzību, aproksimējot datus no tabulām:[3]

$$\gamma_{Zn} = \rho_{Zn} c_{Zn}, \qquad \rho_{Zn} = 7130\ \mathrm{kg/m^3} \tag{12}$$

	0 °C	20 °C	200 °C	300 °C	400 °C	420 °C	
k_{Zn}		119	113	108		100	$\mathrm{W/(m \cdot K)}$
c_{Zn}	381		415		431		$\mathrm{J/(kg \cdot K)}$
γ_{Zn}	2745050		2980340		3251280		$\mathrm{J/(m^3 \cdot K)}$



k_{Zn}

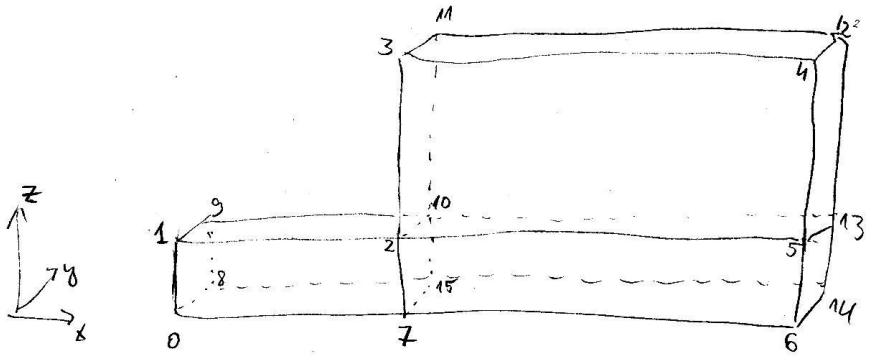


γ_{Zn}

[3] *VDI-Wärmeatlas*. 9. Aufl. Berlin : Springer, 2002

Režģa ģenerēšana

Ģeometriju, izmērus un režģa ģenerēšanas parametrus norādām datnē *constant/polyMesh/blockMeshDict*. Apgabalu brīvi sanumurējam un pēc tā veidojam režģa aprakstu.



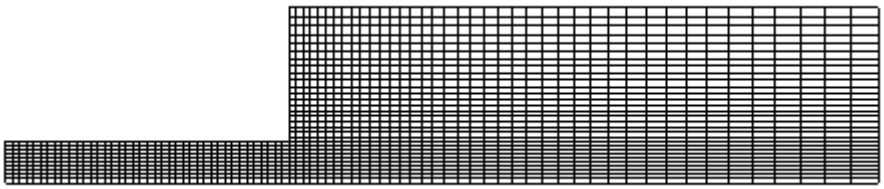

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

convertToMeters 0.001;

vertices
(
    (0  0   0  )
    (0  0   1.9)
    (13 0   1.9)
    (13 0   8  )
    (40 0   8  )
    (40 0   1.9)
    (40 0   0  )
    (13 0   0  )
    (0  0.2 0  )
    (0  0.2 1.9)
    (13 0.2 1.9)
    (13 0.2 8  )
    (40 0.2 8  )
    (40 0.2 1.9)
    (40 0.2 0  )
    (13 0.2 0  )
);

blocks
(
    hex (0 7 15 8 1 2 10 9) (40 5 10) simpleGrading (1 1 1)
    hex (7 6 14 15 2 5 13 10) (40 5 10) simpleGrading (4 1 1)
    hex (2 5 13 10 3 4 12 11) (40 5 20) simpleGrading (4 1 2)
);

patches
(
    patch fuseElement     //drošinātāja elements - tievākā daļa
    (
        (1 2 10 9)
        (8 9 10 15)
    )
    patch fuseBlade     //drošinātāja asmeņi
    (
```

```
            (11 12 13 10)
            (10 13 14 15)
            ( 2  3 11 10)
            ( 3  4 12 11)
        )
        symmetryPlane fuseSymmetry
        (
            (0 7 2 1)
            (2 5 4 3)
            (2 7 6 5)
            (0 1 9 8)
            (4 5 13 12)
            (5 6 14 13)
            (0 8 15 7)
            (6 7 15 14)
        )
    );
```

Patch virsmu normālēm jābūt vērstām uz iekšpusi. Tas tā ir, ja, skatoties no iekšpuses, punkti sarindoti pulksteņrādītāja virzienā. Līdzīgi ar blokiem – pirmās virsmas normālei jābūt vērstai uz iekšu. Pirmo virsmu definē bloka pirmās četras virsotnes.

Ģenerējam režģi, piemēra direktorijā *fuseCartesian* izpildot komandu *blockMesh*, ko varam pēc tam apskatīties ar programmu ParaView.

```
blockMesh
```



Fizikālās īpašības

constant/electricProperties – metāla elektriskās īpašības

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      electricProperties;
}

I               I [ 0 0 0 0 0 1 0 ] 75.;  // 150% strāva no nominālvērtības

rhoRef          rhoRef [1 3 -3 0 0 -2 0] 5.95e-8;

alphaRef        alphaRef [0 0 0 -1 0 0 0] 4.20e-3;

TRef            TRef [0 0 0 1 0 0 0] 20.0;
```

Aprēķinus veiksim pie dažādām strāvas stipruma I vērtībām, ko mainīsim šajā datnē.

constant/A.org – šķērsgriezuma laukums

Norādām šķērsgriezuma laukumu, ko programma izmantos, aprēķinot siltuma ģenerēšanas jaudu. Datnē ierakstām vērtību „resnākajam" apgabalam, bet tievākajam apgabalam to vēlāk norādīsim ar utilītprogrammas *setFields* palīdzību.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      A;
}

dimensions      [0 2 0 0 0 0 0];

internalField   uniform 6.4e-6;

boundaryField
{
    fuseElement
    {
        type            fixedValue;
        value           uniform 6.4e-6;
    }
    fuseBlade
    {
        type            fixedValue;
        value           uniform 6.4e-6;
    }
    fuseSymmetry
    {
        type            symmetryPlane;
    }
}
```

constant/CRho0 –īpatnējās siltumietilpības polinoma brīvais loceklis

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      CRho0;
}

dimensions      [1 -1 -2 -1 0 0 0];

internalField   uniform 2.74505e6;

boundaryField
{
    fuseElement
    {
        type            fixedValue;
        value           uniform 2.74505e6;
    }
    fuseBlade
    {
        type            fixedValue;
        value           uniform 2.74505e6;
    }
    fuseSymmetry
    {
        type            symmetryPlane;
    }
}
```

Datnes *CRho1* un *CRho2*, kas raksturo pārējos polinoma koeficientus, izveido līdzīgi: 1) nomaina atbilstošā objekta nosaukumu; 2) samazina temperatūras dimensiju ar katru nākamo datni par vienu kārtu (ceturtā vērtība dimensiju sarakstā); 3) vērtības ņem no formulas (10).

constant/K0 – īpatnējā siltumvadāmība

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      K0;
}

dimensions      [1 1 -3 -1 0 0 0];

internalField   uniform 119.477;

boundaryField
{
    fuseElement
    {
        type            fixedValue;
        value           uniform 119.477;
    }
    fuseBlade
    {
        type            fixedValue;
        value           uniform 119.477;
    }
    fuseSymmetry
    {
        type            symmetryPlane;
    }
}
```

Datnes *K1*, *K2* un *K3* izveido līdzīgi, vērtības ņemot no formulas (9).

Sākuma nosacījumi

Datnē *0/T* norādīsim drošinātāja sākuma temperatūru, kas būs vienāda ar apkārtējās vides temperatūru $T_\infty = 65°\mathrm{C}$, kā arī robežnosacījumus, kas ietver nelineāru siltumapmaiņas koeficientu:

$$\alpha = \alpha_{conv} + \alpha_{rad}, \quad \alpha_{rad} = \varepsilon\sigma(\tilde{T}^2 + \tilde{T}_\infty^2)(\tilde{T} + \tilde{T}_\infty), \quad \sigma = 5.6704 \cdot 10^{-8}\ \mathrm{W/(m^2\,K^4)} \quad (13)$$

Temperatūra ar tildes zīmi $\tilde{T}$ jārēķina Kelvina grādos. Virsmas siltumstarojuma koeficientu sākotnēji ņemam $\varepsilon = 0.05$, bet konvektīvā siltumapmaiņas koeficienta α_{conv} aprēķināšanai izmantojam citu publikāciju datus.[4] [5] Tā kā drošinātāja atrašanās pret zemi precīzi nav zināma, izmantojam formulas, kas paredzētas cilindriskam ķermenim, saglabājot to pašu plūsmas garumu (objekta apkārtmēru).

---

[4] *VDI-Wärmeatlas*. 9. Aufl. Berlin : Springer, 2002

vai Incropera F.P. *Fundamentals of Heat and Mass Transfer*. 6th ed. Wiley, 2006. 1024 p.

[5] Ilgevicius A. *Analytical and Numerical Analysis and Simulation of Heat Transfer in Electrical Conductors and Fuses.* Neubiberg : Universitaet der Bundeswehr Muenchen, 2004. Dissertation.

$$T = (T_{face} + T_{\infty})/2$$

$$l = 2(b + h_i), \quad (h_0 = h,\ h_1 = H)$$

$$\lambda = 2.434\cdot10^{-2} + 7.658\cdot10^{-5}\,T - 4.030\cdot10^{-8}\,T^2 + 3.101\cdot10^{-11}\,T^3 - 1.065\cdot10^{-14}\,T^4$$

$$\nu = 1.352\cdot10^{-5} + 8.876\cdot10^{-8}\,T + 1.098\cdot10^{-10}\,T^2 - 4.035\cdot10^{-14}\,T^3 + 1.347\cdot10^{-17}\,T^4$$

$$\mathrm{Pr} = 0.7109 - 1.578\cdot10^{-4}\,T + 5.885\cdot10^{-7}\,T^2 - 6.198\cdot10^{-10}\,T^3 + 2.180\cdot10^{-13}\,T^4$$

$$\beta = 0.993048/(\mathrm{T}+270.127)$$

$$Gr = g\beta l^3 |T_{face} - T_{\infty}| / \nu^2$$

$$Ra = Gr \cdot \mathrm{Pr}$$

$$f = \left(1 + (0.559/\mathrm{Pr})^{9/16}\right)^{-16/9}$$

$$Nus = \left(0.752 + 0.387(Ra \cdot f)^{1/6}\right)^2$$

$$\alpha_{conv} = Nus \cdot \lambda / l$$

```
  FoamFile
  {
      version     2.0;
      format      ascii;
      class       volScalarField;
      object      T;
  }
  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

  dimensions      [0 0 0 1 0 0 0];

  internalField   uniform 65;

  boundaryField
  {
      fuseElement
      {
          type            groovyBC;
          value           uniform 0;
          variables       "
h=1.9e-3;
b=0.2e-3;
epsilon=0.05;
Tinf=65;
L=2*(h+b);
TT=(T+Tinf)/2;
g=9.81;
sigma=5.6704e-8;
lambda=2.4340e-2+7.6575e-5*TT-4.0303e-8*pow(TT,2)+3.1098e-11*pow(TT,3)
       -1.0648e-14*pow(TT,4);
nu=1.3520e-5+8.8761e-8*TT+1.0979e-10*pow(TT,2)-4.0352e-14*pow(TT,3)
       +1.3466e-17*pow(TT,4);
pr=0.71091-1.5775e-4*TT+5.8848e-7*pow(TT,2)-6.1980e-10*pow(TT,3)
       +2.1796e-13*pow(TT,4);
beta=0.993048/(TT+270.127);
gr=g*beta*mag(T-Tinf)*pow(L,3)/pow(nu,2);
ra=gr*pr;
f=pow(1+pow(0.559/pr,0.5625),-1.778);
nus=pow(0.752+0.387*pow(ra*f,0.1667),2);
alphaConv=nus*lambda/L;
alphaRad=epsilon*sigma*(sqr(T+273.15)+sqr(Tinf+273.15))*(T+Tinf+2*273.15);
alpha=alphaConv+alphaRad;
C=K/alpha/mag(delta());";
          valueExpression "Tinf";
          fractionExpression "1/(1+C)";
      }
```

```
      fuseBlade
      {
          type            groovyBC;
          value           uniform 0;
          variables       "
h=8.0e-3;
b=0.2e-3;
epsilon=0.05;
Tinf=65;
L=2*(h+b);
TT=(T+Tinf)/2;
g=9.81;
sigma=5.6704e-8;
lambda=2.4340e-2+7.6575e-5*TT-4.0303e-8*pow(TT,2)+3.1098e-11*pow(TT,3)
       -1.0648e-14*pow(TT,4);
nu=1.3520e-5+8.8761e-8*TT+1.0979e-10*pow(TT,2)-4.0352e-14*pow(TT,3)
       +1.3466e-17*pow(TT,4);
pr=0.71091-1.5775e-4*TT+5.8848e-7*pow(TT,2)-6.1980e-10*pow(TT,3)
       +2.1796e-13*pow(TT,4);
beta=0.993048/(TT+270.127);
gr=g*beta*mag(T-Tinf)*pow(L,3)/pow(nu,2);
ra=gr*pr;
f=pow(1+pow(0.559/pr,0.5625),-1.778);
nus=pow(0.752+0.387*pow(ra*f,0.1667),2);
alphaConv=nus*lambda/L;
alphaRad=epsilon*sigma*(sqr(T+273.15)+sqr(Tinf+273.15))*(T+Tinf+2*273.15);
alpha=alphaConv+alphaRad;
C=K/alpha/mag(delta());";
          valueExpression "Tinf";
          fractionExpression "1/(1+C)";
      }


      fuseSymmetry
      {
          type            symmetryPlane;
      }
  }
```

Uzskatāmības dēļ mainīgie šeit izvietoti viens zem otra, bet datnē visi jāsaraksta aiz atslēgvārda *variables* vienu aiz otra bez atstarpēm.

Risināšanas parametri

system/controlDict

```
  FoamFile
  {
      version     2.0;
      format      ascii;
      class       dictionary;
      location    "system";
      object      controlDict;
  }

  libs ( "libOpenFOAM.so" "libgroovyBC.so" );

  application     laplacianFoam;

  startFrom       latestTime;

  startTime       0;

  stopAt          endTime;
```

```
endTime         10.;

deltaT          0.01;

writeControl    runTime;

writeInterval   0.5;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
(
    T
    {
        type         volumeMinMax;
        functionObjectLibs ("libsimpleFunctionObjects.so");
        verbose      true;
        fields       (T);
    }

    probes
    {
        name               probes;
        type               probes;
        functionObjectLibs ("libsampling.so");
        probeLocations
        (
            (1.5e-4 1.8e-4 18.0e-4)
            (1.5e-4 1.8e-4  1.0e-4)
        );
        fields             (T);
        outputControl      timeStep;
        outputInterval     1;
     }
);
```

Ar *functionObjects* funkcionalitātes palīdzību katrā laika solī tiks saglabāta minimālā un maksimālā temperatūra, kā arī temperatūra divos punktos. Tas tiek panākts ar funkciju *volumeMinMax*, kas iekļauts papildinājumā *swak4foam*, un funkciju *probes*, kas nāk līdzi OpenFoam pakotnei. Katru parādīto funkciju objektu var lietot atsevišķi. Viens no norādītajiem zondes punktiem atrodas tievākajā apgabalā tuvāk centram, otrs – pie ārējās malas, lai pārliecinātos, ka kušanas temperatūra sasniegta visā šķērsgriezumā.

constant/fvSchemes

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
```

```
    object      fvSchemes;
}

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(T)         Gauss linear;
}

divSchemes
{
    default         none;
}

laplacianSchemes
{
    default         none;
    laplacian(K,T)  Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    T               ;
}
```

constant/fvSolutions

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}

solvers
{
    T
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 4;
}
```

Lai arī parametrs *nNonOrthogonalCorrectors* domāts papildu iterāciju veikšanai neortogonālu režģu gadījumos, mums tas noder, pārrēķinot nelineāro fizikālo koeficientu vērtības.

Sākotnēja lauku modificēšana

No datnes *A.org* izveidosim un modificēsim datni *A*, ietverot drošinātāja tievā apgabala šķērsgriezuma laukumu. Palīgā ņemsim utilītprogrammu *setFields*, kurai parametrus norādam datnē setFieldsDict.

constant/setFieldsDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}

regions
(
    boxToCell
    {
        box (0 0 0) (13e-3 1 1);            //Apgabals, kurā ietilpst tikai
        fieldValues                         //drošinātāja tievā daļa
        (
            volScalarFieldValue A     1.52e-6
        );
    }
);
```

setValues

Izveidosim datni *setValues*:

```
cd constant
cp A.org A
cd ..
setFields -constant
```

un pēc tam no komandrindas to pārveidosim par izpildāmo un palaidīsim:

```
chmod +x setValues
./setValues
```

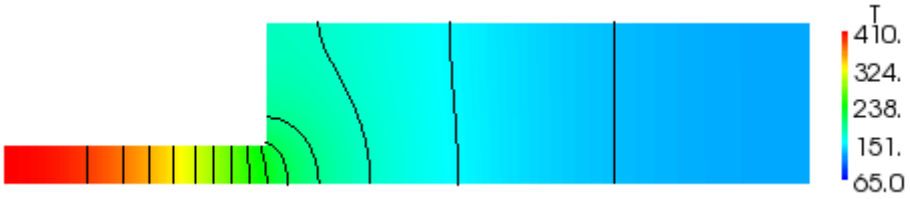
Iegūto rezultātu var apskatīt datnē *A* un salīdzināt ar *A.org*.

4. Aprēķini un rezultāti

Palaižam aprēķinus, izpildot komandu *jouleHeatingFoam*:

```
jouleHeatingFoam
```

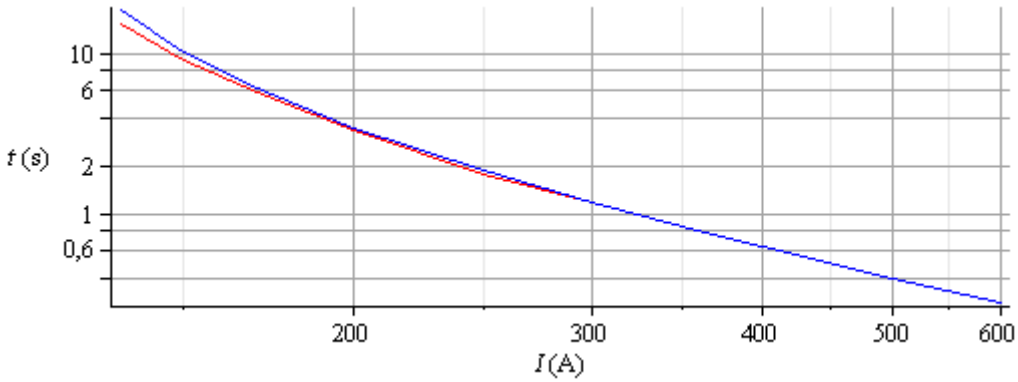
Kad skaitļošana beidzas, *datnē probes/0/T* vai *volumeMinMax_T/0/T* konstatējam, ka cinka kušanas temperatūra, kas ir 420 grādi pēc Celsija, tiek sasniegta 9.4 sekundēs. Temperatūras sadalījumu 9. sekundē apskatāmies ar programmu ParaView.



Kontūrlīnijas iegūtas ar filtru *Filters* > *Common* > *Contour*. Temperatūras starpība starp tām ir 20 grādi, kas norādīts filtra *Object Inspector* sadaļā *Properties* > *Isosurfaces*, bet melnu līniju iegūšanai – sadaļā *Display* > *Style* izvēlēts *Representation*: *Surface With Edges*.

Mainot strāvas stiprumu un atbilstoši arī laika soli, varam veikt aprēķinu sēriju un apkopot rezultātus tabulā un grafikā. Salīdzināšanai veicam arī aprēķinus gadījumam, kad $\varepsilon = 0.5$. Fizikāli tas varētu notikt, ja cinka virsma būtu oksidējusies, drošinātājam pirms tam ilgstoši strādājot tuvu kušanas temperatūrai. Grafiks veidots ar programmu *Maple*.

	%	135	150	170	200	250	300	350	400	500	600
	A	67.5	75	85	100	125	150	175	200	250	300
1.	$\varepsilon = 0.05$	15.8	9.4	5.8	3.4	1.8	1.2	0.84	0.63	0.40	0.28
2.	$\varepsilon = 0.5$	18.8	10.4	6.1	3.5	1.9	1.2	0.84	0.63	0.40	0.28



Materiāls tapis ar ESF projekta nr. 2009/0223/1DP/1.1.1.2.0/APIA/VIAA/008 atbalstu.